DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1. Latar Belakang

Indonesia menempati posisi ketiga secara global dalam jumlah akun yang terdampak kebocoran data, dengan jumlah sekitar 12,7 juta akun yang mengalami kebocoran hingga kuartal III-2022 [1] . Berdasarkan laporan Badan Siber dan Sandi Negara (BSSN) terkait serangan pada aplikasi web tahun 2023, tercatat sebanyak 347 kasus serangan siber, yang terdiri dari 186 serangan pada aplikasi pemerintahan di bidang administrasi, 60 serangan pada sektor lainnya, 38 serangan pada sektor keuangan, 24 serangan pada sektor transportasi, 18 serangan pada sektor ESDM, 5 serangan pada sektor TIK, 5 serangan pada sektor kesehatan, 4 serangan pada sektor pangan, dan 2 serangan pada sektor pertahanan [2] . Meskipun jumlah serangan pada sektor lain relatif sedikit, keamanan aplikasi di bidang pendidikan tetap perlu mendapat perhatian serius. Hal ini disebabkan karena tanpa adanya sistem keamanan yang memadai, data perusahaan atau instansi akan lebih rentan diretas, sehingga berpotensi menurunkan kredibilitas dan citra institusi tersebut [3]. Oleh karena itu, penerapan keamanan aplikasi sejak tahap awal pengembangan akan lebih efektif dan hemat biaya dibandingkan dengan penanganan setelah terjadinya serangan, mengingat seiring waktu akan bermunculan permasalahan baru yang sebelumnya belum teridentifikasi [4].

Perusahaan dan instansi dari berbagai industri semakin memanfaatkan perkembangan teknologi dengan menciptakan aplikasi sebagai platform untuk terhubung dengan pengguna [4] . Salah satu contohnya adalah aplikasi S.E.E.D.S *(Student Enrollment and Education Data System)* milik Telkom University. Aplikasi berbasis web ini memudahkan mahasiswa baru untuk mendaftar ulang tanpa harus datang langsung ke kampus. Aplikasi S.E.E.D.S ini dibangun menggunakan kerangka kerja NestJS. Dimana kerangka kerja ini merupakan kerangka kerja yang berbasis nodejs dimana biasanya digunakan untuk membangun aplikasi pada sisi server. Keamanan aplikasi menjadi perhatian utama, mengingat data pribadi mahasiswa yang sensitif, seperti KTP, nomor rekening, alamat, dan nama orang tua, perlu dilindungi dengan ketat, terutama pada fitur registrasi yang memuat banyak informasi pribadi. Melalui wawancara yang

dilakukan dengan tim pengembang aplikasi S.E.E.D.S, diperoleh informasi bahwa aplikasi tersebut belum pernah menjalani pengujian keamanan, baik dengan metode OWASP maupun metode pengujian lainnya berita acara dapat dilihat pada [lampiran 1]. Hal ini menjadi perhatian serius mengingat potensi serangan siber yang dapat terjadi kapan saja, dengan dampak yang merugikan bagi privasi pengguna. Oleh karena itu, pengujian keamanan atau kerentanan aplikasi secara menyeluruh merupakan langkah krusial untuk mengidentifikasi potensi serangan sekaligus mencegah pencurian data. Mengingat aplikasi dapat diakses kapan saja dan di mana saja selama terhubung ke internet, menjaga keamanan data menjadi prioritas utama agar mahasiswa baru dapat melakukan pendaftaran dengan tenang tanpa kekhawatiran terhadap privasi mereka [3].

Dalam upaya menjaga keamanan aplikasi, diperlukan pengujian keamanan atau kerentanan untuk mengetahui apakah aplikasi memiliki kelemahan yang dapat dimanfaatkan oleh pihak tidak bertanggung jawab [5]. Adapun standar keamanan terkait aplikasi website yang populer diantaranya yaitu SANS/CWE Top 25 dan OWASP. SANS/CWE Top 25 merupakan daftar kelemahan yang biasa terjadi pada perangkat lunak [6] . OWASP *(Open Web Application Security Project)* merupakan organisasi non profit yang bertujuan untuk meningkatkan keamanan suatu aplikasi [7] . OWASP memiliki proyek bernama OWASP Top 10 dimana proyek ini merupakan daftar dari 10 serangan yang paling kritis pada keamanan aplikasi web [8]. Perbedaan dari OWASP dan SANS/CWE Top 25 yaitu OWASP lebih berfokus pada kerentanan atau serangan yang biasa terjadi pada aplikasi web sedangkan SANS/CWE Top 25 tidak berfokus pada aplikasi web saja melainkan kepada aplikasi non web juga seperti desktop, IoT dan lainnya di luar web. Pemilihan metode OWASP untuk pengujian keamanan aplikasi S.E.E.D.S didasarkan pada fokus utamanya pada aplikasi web dan tidak akan mengembangkkan aplikasi S.E.E.D.S selain aplikasi berbasis website[lampiran 1], yang sesuai dengan karakteristik S.E.E.D.S sebagai aplikasi berbasis web. Hal ini juga menjadi keunggulan OWASP dibandingkan standar lain seperti SANS/CWE Top 25, yang mencakup kelemahan perangkat lunak secara umum dan tidak secara spesifik berfokus pada aplikasi web [9] . Selain itu, jika dibandingkan, hanya sekitar delapan dari daftar SANS/CWE Top 25 yang relevan dengan

keamanan aplikasi web, sehingga penggunaan OWASP Top 10 sebagai dokumen pengujian aplikasi web menjadi pilihan yang lebih tepat [9] . OWASP digunakan dalam penelitian ini karena merupakan metode keamanan yang secara khusus memfokuskan pada aplikasi berbasis web. Mengingat S.E.E.D.S juga merupakan aplikasi web, penggunaan metode OWASP menjadi pilihan yang tepat dan relevan untuk mengidentifikasi serta mengatasi potensi kerentanan yang mungkin terjadi pada aplikasi tersebut. Dengan demikian, menggunakan OWASP Top 10 sebagai acuan pengujian keamanan aplikasi S.E.E.D.S adalah langkah terbaik untuk memastikan aplikasi diuji sesuai dengan standar yang paling relevan dan serangan yang paling umum terjadi.

Penelitian ini bertujuan menguji keamanan aplikasi S.E.E.D.S menggunakan panduan *OWASP Top 10* serta memberikan evaluasi dan rekomendasi mitigasi berdasarkan dokumen tersebut apabila ditemukan kerentanan. Hasil penelitian diharapkan dapat menunjukkan tingkat kepatuhan aplikasi S.E.E.D.S Telkom University terhadap standar keamanan *OWASP*, menjadi acuan peningkatan keamanan aplikasi berbasis web, serta mengidentifikasi peran kerangka kerja NestJS dalam melindungi keamanan aplikasi.

1.2. Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana cara mengidentifikasi kerentanan pada aplikasi S.E.E.D.S berdasarkan panduan *OWASP Top 10*?
2. Bagaimana rekomendasi mitigasi yang dapat diterapkan terhadap kerentanan yang ditemukan berdasarkan panduan *OWASP Top 10* pada aplikasi S.E.E.D.S?

1.3. Tujuan dan Manfaat

1. Mengetahui hasil identifikasi kerentanan pada aplikasi S.E.E.D.S berdasarkan panduan *OWASP Top 10*.
2. Mengetahui rekomendasi mitigasi terhadap kerentanan yang ditemukan setelah pengujian berdasarkan panduan *OWASP Top 10* pada aplikasi S.E.E.D.S.

1.4. Batasan Masalah

Penelitian ini akan melakukan pengujian keamanan pada aplikasi S.E.E.D.S dengan mengacu pada standar dokumentasi *OWASP Top 10*. Pengujian difokuskan untuk mengidentifikasi potensi kerentanan, kemudian memberikan rekomendasi mitigasi perbaikan apabila ditemukan celah keamanan, di mana rekomendasi tersebut sepenuhnya didasarkan pada pedoman *OWASP Top 10*. Selain itu, penelitian ini akan melakukan *literature review* untuk mengkaji sejauh mana kerangka kerja NestJS berpengaruh terhadap aspek keamanan aplikasi.

Batasan penelitian ini terletak pada ruang lingkup pengujian, yang hanya mencakup dua fitur utama pada aplikasi S.E.E.D.S yang digunakan oleh calon mahasiswa, yaitu **fitur registrasi ulang** dan **fitur undur diri**. Pemilihan kedua fitur ini didasarkan pada pertimbangan bahwa keduanya memproses, menyimpan, dan mengelola data pribadi mahasiswa, seperti identitas, informasi akademik, dan dokumen pendukung yang bersifat sensitif. Data pribadi tersebut memiliki risiko tinggi jika terekspos atau disalahgunakan, sehingga pengujian keamanan pada fitur-fitur ini menjadi prioritas utama. Dengan pembatasan ruang lingkup tersebut, penelitian ini diharapkan dapat menghasilkan evaluasi yang terfokus, mendalam, dan memberikan rekomendasi yang relevan untuk meningkatkan keamanan aplikasi S.E.E.D.S, khususnya pada fitur yang memiliki dampak besar terhadap perlindungan data pribadi mahasiswa.

1.5. Jadwal Pelaksanaan

Tabel 1.1 Jadwal Pelaksanaan Tugas Akhir

No.	Deskripsi Tahapan	Bulan 1	Bulan 2	Bulan 3	Bulan 4	Bulan 5	Bulan 6
1	Analisis kebutuhan						
2	Perencanaan pengujian						
3	Pembuatan skenario uji						
4	Pengaturan lingkungan uji						
5	Eksekusi skenario uji						
6	Evaluasi hasil pengujian						

7	Penyusunan Laporan/Buku TA						

BAB 2 TINJAUAN PUSTAKA

2.1. OWASP

OWASP merupakan organisasi non profit yang bertujuan untuk meningkatkan keamanan aplikasi[7]. Visi dari *OWASP* sendiri adalah "Tidak ada lagi perangkat lunak yang tidak aman" dengan misi "Menjadi komunitas terbuka global yang mendorong perangkat lunak aman melalui edukasi, alat, dan kolaborasi." dari visi misi ini diketahui memang *OWASP* berfokus pada keamanan perangkat lunak. Dan sebagai langkah untuk mewujudkan visi dan misi ini, *OWASP* memiliki banyak proyek yang bertujuan untuk membantu para pengembang dan komunitas dalam mengamankan aplikasi. Proyek dari *OWASP* untuk mengamankan aplikasi berbasis web adalah proyek *OWASP* Top 10, dimana dalam dokumen *OWASP* Top 10 ini terdapat daftar serangan yang kritis dan sering terjadi pada aplikasi web sehingga menggunakan dokumentasi ini untuk meminimalkan resiko akan terjadinya kerentanan pada aplikasi web adalah langkah pertama dan efektif untuk menjaga aplikasi web terhadap serangan yang sering terjadi [8].

Adapun daftar dari dokumen Top 10 *OWASP* adalah :

1. A01: *Broken Access Control*

   Kerentanan pada website dimana seorang pengguna bisa melakukan aksi diluar dari hal yang sudah ditetapkan pada aplikasi sebelumnya[8].

2. A02: *Cryptographic Failures*

   Kerentanan pada aplikasi dimana aplikasi tidak melindungi data pada saat transit atau saat disimpan. Misalnya data kata sandi, nomor kartu kredit, informasi pribadi, dan data yang disebutkan pada peraturan undang-udang setempat[8].

3. A03: *Injection*

   Kerentanan pada aplikasi dimana aplikasi bisa mengeksekusi kode program atau perintah berdasarkan input pengguna melalui aplikasi. Contohnya aplikasi mengeksekusi perintah SQL berdasarkan masukan pengguna dimana seharusnya masukan dari pengguna hanyalah sebuah data biasa bukan merupakan perintah SQL yang valid[8].

4. A04: *Insecure Design*

   Kerentanan pada aplikasi dimana aplikasi tidak dipertimbangkan keamanannya sejak awal[8].

5. A05: *Security Misconfiguration*

   Kerentanan pada aplikasi dimana konfigurasi aplikasi masih salah contohnya aplikasi menampilkan pesan *error* yang sangat detail atau aplikasi masih menyimpan akun testing atau akun bawaan[8].

6. A06: *Vulnerable and Oudated Components*

   Kerentanan pada aplikasi dimana aplikasi menggunakan pustaka pihak ketiga yang sudah usang atau lama[8].

7. A07: *Identification and Authentication Failures*

   Kerentanan pada proses login pada aplikasi dimana aplikasi bisa dilakukan *brute force* atau aplikasi masih menyimpan akun pengguna bawaan[8].

8. A08: *Software and Data Integrity Failures*

   Kerentanan pada aplikasi dimana aplikasi menggunakan pustaka pihak ketiga yang berasal dari sumber yang tidak dipercaya[8].

9. A09: *Security Logging and Monitoring Failures*

   Kerentanan pada aplikasi dimana aplikasi tidak melakukan sama sekali *log* saat proses transaksi seperti *login* atau saat *log* aplikasi menampilkan data yang sensitif[8].

10. A10: *Server Side Request Forgery*

    Kerentanan pada aplikasi dimana aplikasi melakukan pemanggilan ke sumber yang tidak dikenal dari *url* masukan pengguna[8].

2.2. Urgensi Keamanan

Keamanan pada aplikasi penting untuk dilakukan, sudah jelas bahwa keamanan ini penting untuk melindungi data dan aset digital dari akses yang ilegal [3] . Dalam OSI model, aplikasi merupakan layer atau lapisan yang berinteraksi secara langsung oleh pengguna dimana secara otomatis lapisan ini yang paling dekat dengan penyerang dimana penyerang dapat dilakukan oleh siapapun dan menargetkan siapa saja[3]. Dalam lingkup bisnis dan organisasi keamanan penting untuk dijaga karena jika data-data perusahaan rentan untuk diretas, maka akan mencoret kredibilitas dan citra diri perusahaan atau organisasi dan juga

memastikan keamanan aplikasi dari awal akan lebih murah dibandingkan dengan tidak memastikan keamanan dari awal tentu seiring waktu akan banyak masalah yang baru saja ditemukan[3]. Aplikasi yang memiliki 1 kerentanan sudah disebut aplikasi yang rentan[10] oleh karena itu pengujian keamanan aplikasi diperlukan untuk mengetahui apakah aplikasi memiliki kerentanan atau tidak, jika ada kerentanan walau 1 kerentanan saja maka kemungkinan besar data-data aplikasi akan bisa bocor. Dalam pengujian keamanan aplikasi, ada beberapa macam jenis pengujian keamanan aplikasi, diantaranya adalah:

1. *Dynamic Application Security Testing (DAST)*

Merupakan suatu metode testing untuk mengetahui kerentanan pada aplikasi web, mobile atau api. Dimana metode testing ini menguji keamanan aplikasi yang sedang berjalan dengan cara meniru atau mensimulasikan serangan untuk dicoba diserang kedalam aplikasi. Metode ini terkadang disebut dengan black box testing karena pengujian aplikasi tidak mengetahui kode sumber dari suatu aplikasi. Banyak dari perusahaan menggunakan daftar OWASP Top 10 sebagai patokan atau panduan dalam mengetahui kelemahan pada aplikasinya. [11]

2. *Static Application Security Testing (SAST)*

Merupakan kebalikan daripada DAST, dimana pengujian ini lebih menguji atau berfokus pada kode sumber daripada aplikasi. Dimana pengujian menggunakan metode SAST ini biasanya sudah dilakukan pada tahap development dari pada aplikasi.[12]

3. *Penetration Testing*

Penetration testing atau biasa disingkat dengan pentest merupakan metode pengujian keamanan aplikasi dengan cara melakukan tiruan dari serangan hacker lalu serangan tersebut diuji pada sistem komputer untuk mengetahui kerentanan yang ada. Sekilas hampir sama dengan DAST, namun tujuan dari pentest ini lebih luas dibandingkan dengan DAST. DAST hanya berfokus pada layer atau lapisan aplikasi sedangkan pentest menguji keamanan tidak hanya dari layer atau lapisan aplikasi saja melainkan juga dari sisi jaringan, perangkat keras sampai melakukan pengujian pada pegawai suatu organisasi. Walaupun hampir sama dengan cara melakukan simulasi serangan dari hacker, namun jangkauan dari pentest ini lebih

luas. Pada pentest dibagian aplikasi pun, para tester biasanya menggunakan OWASP Top 10 sebagai patokan atau standar untuk menguji keamanan dari aplikasi[13]. Dalam penelitian ini, peneliti akan menggunakan metode pengujian berjenis DAST karena peneliti hanya berfokus untuk mencari tahu celah keamanan pada aplikasi S.E.E.D.S.

4. *White Box Testing*

White Box Testing merupakan pengujian yang berfokus pada sumber dari aplikasi. Tujuan daripada *testing* ini adalah untuk mengetahui kompleksitas daripada kode program aplikasi, selain itu pengujian ini juga bertujuan untuk mengetahui apakah aplikasi sudah memenuhi atau mengikuti dari desain atau kebutuhan fungsional yang sudah ditetapkan pada saat perancangan dan juga untuk mengetahui apakah pada kode terdapat kerentanan atau tidak [14].

5. *Black Box Testing*

Black Box Testing merupakan pengujian dimana penguji tidak mengetahui internal atau hal teknis daripada aplikasi itu sendiri, artinya penguji hanya mengetahui aplikasi dari luar saja atau aplikasi yang sudah jadi saja tanpa tahu apa yang ada di belakang aplikasi seperti kode aplikasi, *server* aplikasi dan sebagainya [14]

6. *Gray Box Testing*

Gray Box Testing merupakan kombinasi dari *Black Box Testing* dan *White Box Testing* dimana penguji dilakukan dari *internal* aplikasi seperti kode sumber aplikasi dan dari luar aplikasi itu sendiri, sehingga pengujian menggabungkan persepsi penyerang dari luar dan penyerangan orang dalam [14].

2.3. SEEDS

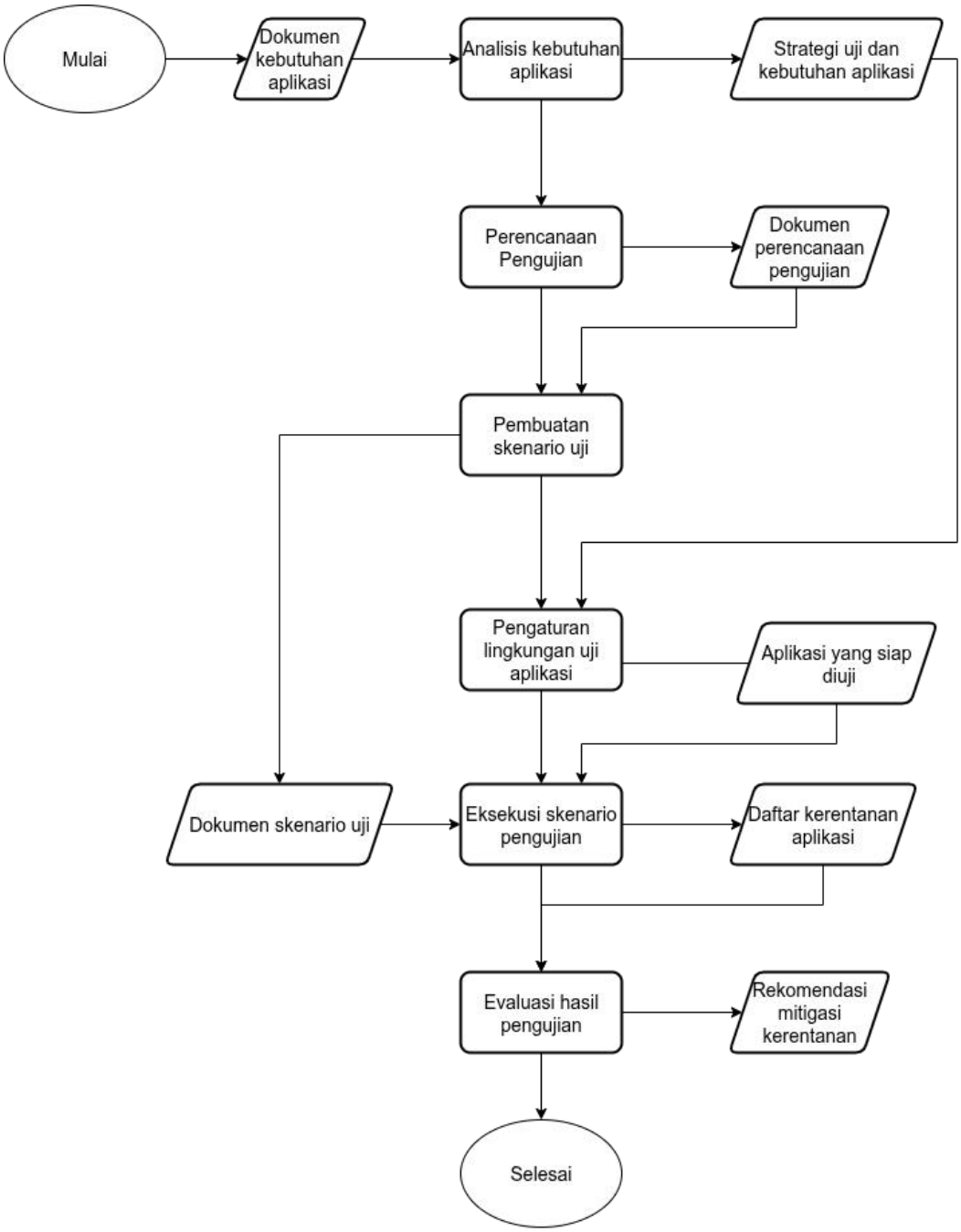
S.E.E.D.S merupakan aplikasi berbasis web dari Telkom University yang memiliki fungsi sebagai alat untuk melakukan registrasi ulang para mahasiswa baru yang akan melanjutkan studi di Telkom University. Aplikasi berbasis web merupakan sistem yang diunggah dan digunakan oleh pengguna via internet, dimana pengguna mengakses web ini menggunakan aplikasi peramban seperti google chrome, firefox, edge, dan sebagainya [15] . Aplikasi kritis dalam bidang komputer sains merujuk pada semua komputer atau jaringan jika tidak tersedia maka berpengaruh besar pada kemampuan operasi suatu organisasi [16] . Maka dari itu aplikasi kritis sangat butuh pemeliharaan yang lebih karena sifat dari aplikasi itu sendiri memiliki pengolahan informasi yang penting [16] . S.E.E.D.S termasuk kedalam kategori aplikasi kritis, dikarenakan jika aplikasi S.E.E.D.S ini mengalami gangguan dan informasi yang berada pada aplikasi S.E.E.D.S tidak dijaga, maka akan mengakibatkan gangguan operasional Telkom University dan reputasi dari Telkom University pun bisa jadi turun[3].

2.4. *Literature Review*

Literature review menunjukkan berbagai pendekatan untuk mengatasi kerentanan aplikasi web dengan mengacu pada standar keamanan OWASP Top 10. Penelitian Vallabhaneni et al [4] menunjukkan efektivitas MobileNet dalam membangun firewall yang mampu menganalisis dan memetakan ancamannya sesuai dengan kategori OWASP Top 10. Hal ini membuat penegasan pada posisi OWASP sebagai pedoman penting dalam pengujian keamanan aplikasi. Berbeda dengan Mohamed Mohideen et al[17] yang mengatakan, dimana pada penelitiannya dapat memperluas wawasan tentang kerentanan plugin WordPress terhadap serangan XSS, sekaligus membuat langkah mitigasi sebagai referensi praktis, sementara itu R.A. Khan et al[11]. menekankan perlunya pengujian keamanan sejak tahap awal pengembangan perangkat lunak untuk menghindari ancaman seperti broken access control dan juga injection. Penelitian lain oleh Lokesh Raju et al [18]. dan Petranović et al [10] . Dengan hasil penelitian memperkuat relevansi OWASP dengan menguji efektivitas firewall ModSecurity dan yang menunjukkan bahwa terdapat banyak aplikasi web yang gagal memenuhi standar OWASP, yang

menandakan bahwa standar ini tetap relevan untuk mengidentifikasi keamanan pada sebuah aplikasi web. Choiriyah et al [5] meneliti aplikasi kesehatan milik pemerintah Indonesia menggunakan pendekatan OWASP Top 10 untuk mengidentifikasi kerentanan dan memberikan solusi mitigasi yang komprehensif. Fredj et al [19] . dan Chillur et al [20] . Menambahkan wawasan penting melalui analisis pola serangan dan langkah perlindungan, menekankan pentingnya pendekatan berbasis standar untuk mengurangi risiko serangan yang kompleks. Anantharaman et al [21] . secara spesifik mengeksplorasi kategori A9 OWASP, menunjukkan metode sistematis untuk mendeteksi dan menangani komponen perangkat lunak yang usang atau rentan. Yin dan Lee [22] , dengan pendekatan yang berbeda, membandingkan keamanan aplikasi berbasis Java dan PHP, memberikan gambaran tentang tingkat risiko terhadap serangan seperti XSS dan SQL injection. Walaupun semua penelitian mengacu pada OWASP Top 10 sebagai panduan standar keamanan, tetapi fokus utamanya cenderung pada pencegahan keamanan di sisi network, atau melakukan literature review untuk menemukan kerentanan yang sering terjadi. Namun beberapa penelitian juga melakukan pengujian keamanan aplikasi langsung berdasarkan OWASP Top 10. Sebaliknya, penelitian yang akan peneliti lakukan berfokus pada analisis keamanan aplikasi berbasis kerangka kerja nestjs dalam aplikasi S.E.E.D.S. Penelitian ini tidak hanya bertujuan untuk menguji kerentanan aplikasi sesuai OWASP Top 10, tetapi juga untuk menganalisis pengaruh penggunaan NestJS terhadap keamanan aplikasi secara keseluruhan.

BAB 3 METODE PENELITIAN



Gambar 3.1 Alur Penelitian

3.1. Analisis Kebutuhan

Analisis kebutuhan ini akan dilakukan tahapan untuk mengidentifikasi fitur-fitur utama dalam aplikasi S.E.E.D.S, seperti autentikasi pengguna, data apa saja yang

diinputkan dan bagaimana aplikasi berinteraksi dengan *server*. Fitur-fitur ini kemudian dilakukan proses analisis untuk menentukan potensi kerentanannya berdasarkan standar dokumena OWASP Top 10. Proses ini bertujuan untuk memahami skenario-skenario serangan yang relevan dengan aplikasi dan menyusun ruang lingkup pengujian berdasarkan potensi risiko keamanan yang paling signifikan. Hasil dari tahap ini merupakan pemahaman mendalam mengenai OWASP dam fitur-fitur aplikasi S.E.E.D.S menjadi dasar dalam penyusunan skenario uji yang akan dilakukan pada tahap pengujian selanjutnya.

3.2. Perencanaan Pengujian

Perencanaan pengujian dilakukan untuk penyusunan strategi pengujian keamanan pada aplikasi S.E.E.D.S. Pengujian akan dilakukan secara manual dengan pendekatan pengujian *gray-box* karena pengujian akan dilakukan langsung menggunakan antar muka aplikasi dan melalui *api* yang tersedia pada aplikasi namun tanpa mengakses atau menganalisis sumber kode aplikasi. Pengetahuan mengenai OWASP dan fitur yang akan diuji pada tahap sebelumnya menjadi dasar untuk perencanaan pembuatan skenario dan cara menguji aplikasi. Proses dari tahap ini yaitu untuk menentukan strategi pengujian, metode dan alat yang dibutuhkan. Aplikasi pembantu dalam melakukan pengujian Adapun alat untuk membantu dalam proses pengujian aplikasi ini adalah:

Tabel 3.1 Daftar Alat Untuk Membantu Pengujian

Pengujian	**Alat**
A01: Broken Access Control	*Browser devtools, burp suite, webwolf*
A02: Cryptographic Failures	*Burp suite, database gui/cli*
A03: Injection	*Database gui/cli*
A04: Insecure Design	Dokumen perencanaan aplikasi
A05: Security Misconfiguration	*Browser devtools, teks editor*
A06: Vulnerable and Outdated Components	*OWASP depedency check*

A07: Identification and Authentication Failures	*Burp suite*
A08: Software and Data integrity Failures	*Browser devtools, terminal/cmd*
A09: Security Logging and Monitoring Failures	*terminal/cmd*
A10: Server-Side Request Forgery	*Browser devtools*

3.3. Pembuatan Skenario Uji

Skenario pengujian didapatkan berdasarkan contoh penyerangan yang diberikan oleh *OWASP* dan juga melalui aplikasi *OWASP webgoat* yaitu aplikasi dari *OWASP* untuk latihan langsung pengujian berdasarkan *OWASP* Top 10[8] [23] . Rencana pengujian dari tahap sebelumnya akan ditulis dalam bentuk dokumen untuk menjadi pegangan dalam pengujian keamanan aplikasi. Berikut merupakan skenario pengujian pada aplikasi berdasarkan OWASP

3.3.1. Pengujian Kerentanan A01: *Broken Access Control*

Tujuan: Memastikan kontrol akses telah diterapkan dengan baik

Langkah Pengujian:

1. Akses halaman yang seharusnya membutuhkan autentikasi tanpa melakukan autentikasi sama sekali.
2. Manipulasi aplikasi dengan memasukan token palsu pada *cookies* atau *localstorage* untuk meniru pengguna seperti sudah melakukan autentikasi.
3. Modifikasi data milik pengguna lain:
    - *Intercept request* pada browser menggunakan *burp suite.*
    - Lihat data yang dikirim, lalu lihat apakah ada data yang terindikasi sebagai tanda pengenal dari pengguna dan coba ubah tanda pengenal tersebut.
    - Kirim ulang request yang telah dimodifikasi.

4. Manipulasi hak akses dengan mengubah isi token menjadi admin.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 2].

3.3.2. Pengujian Kerentanan A02: *Cryptographic Failure*

Tujuan: Memastikan data sensitif telah dienkripsi dan protokol komunikasi aman.

Langkah Pengujian:

1. *Intercept request* menggunakan burp suite lalu ubah request menjadi http untuk melihat apakah bisa aplikasi di request menggunakan koneksi yang tidak aman.
2. Input data sensitif (contoh: *password*, nomor rekening), lalu verifikasi penyimpanannya pada *database* apakah terenkripsi atau *plaintext.*

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 3].

3.3.3. Pengujian Kerentanan A03: *Injection*

Tujuan: Menguji apakah input dapat disisipkan perintah berbahaya.

Langkah Pengujian:

1. Input SQL pada suatu *form* atau *query* param, contoh ada suatu api untuk mengambil data maka akan disisipkan kode sql seperti http://app.com/view?id=';-- - *UNION SELECT*;-- - ' yang berguna untuk mengambil semua data.
2. Input kode javascript pada form, lalu observasi apakah kode js akan di eksekusi pada aplikasi. kode yang diinputkan adalah kode untuk mengambil token autentikasi pada cookies atau localstorage seperti <script>document.cookie</script> atau <script>localstorage</script>.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 4].

3.3.4. Pengujian Kerentanan A04: *Insecure Design*

Tujuan: Mengidentifikasi desain sistem yang tidak aman.

Langkah Pengujian:

1. Lakukan seluruh pengujian A01–A10 untuk menemukan kerentanan pada aplikasi. Contohnya: melakukan logging di tiap aksi pengguna, semisalnya jika aplikasi tidak melakukan logging maka aplikasi memiliki kerentanan pada kategori A09: *Security Logging and Monitoring Failures*, lalu untuk mengetahui apakah aplikasi rentan dia A04: *Insecure Design* adalah dengan melakukan cek pada dokumen perancangan aplikasi apakah ada dokumen mengenai perancangan bagaimana logging akan dibuat. Jika ada maka aplikasi hanya rentan pada A09 saja jika tidak maka termasuk pada A04: *Insecure design.*

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 5].

3.3.5. Pengujian Kerentanan A05: *Security Misconfiguration*

Tujuan: Menilai konfigurasi keamanan sistem.

Langkah Pengujian:

1. Melakukan pengujian seperti input data yang tidak sesuai atau meng nutup koneksi database dengan tujuan mengetahui apakah aplikasi menampilkan pesan *error* yang sangat detail seperti *stack trace error* kode program, *ip address* atau *port database* dan sebagainya yang mana ini merupakan informasi yang bisa dibergunakan penyerang untuk mengakses ke server aplikasi.
2. Melakukan serangan CSRF:
    - Buat halaman login yang serupa dimana form login akan mengirimkan data *email* dan *password* ke aplikasi S.E.E.D.S.
    - Melakukan login pada halaman login yang serupa untuk menyerupai skenario pengguna tertipu pada aplikasi yang palsu, sehingga penyerang mendapatkan token autentikasi.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 6].

3.3.6. Pengujian Kerentanan A06: *Vurnerable and Outdated Components*

Tujuan: Mengidentifikasi *library* pihak ketiga yang rentan.

Langkah Pengujian:

1. Menggunakan OWASP *depedency check* pada projek aplikasi menggunakan perintah *"dependency-check.sh --out . --scan ./"*.
2. Identifikasi hasil OWASP *depedency check* untuk mengetahui apakah ada library pihak ketiga yang rentan atau tidak.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 7].

3.3.7. Pengujian Kerentanan A07: *Identification and Authentication Failures*

Tujuan: Mengidentifikasi perlindungan sistem dari kegagalan proses autentikasi.

Langkah Pengujian:

1. Melakukan *brute force login,* untuk mengetahui apakah sistem akan memblokir *request* jika terjadi request terus menerut dalam waktu yang berdekatan.
2. Simulasi login dari komputer publik:
    - Login tanpa *logout* lalu tutup tab atau *close browser.*
    - Buka kembali aplikasi dalam waktu yang berbeda, apakah aplikasi masih bisa di akses atau tidak.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 8].

3.3.8. Pengujian Kerentanan A08: *Software and Data Integrity Failures*

Tujuan: Memastikan librari pihak ketiga dari sumber terpercaya.

Langkah Pengujian:

1. Menjalankan perintah *"npm config get registry"* apakah aplikasi mengunduh *library* pihak ketiga dari https://registry.npmjs.org
2. Menggunakan *developer alat pengujian* pada *browser* dan *cek library* pada aplikasi website apakah menggunakan *library* dari domain *CDN* yang tidak terpercaya atau tidak.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 9].

3.3.9. Pengujian Kerentanan A09: *Security Logging and Monitoring Failures*

Tujuan: Memverifikasi adanya mekanisme pencatatan aktivitas sistem.

Langkah Pengujian:

1. Melakukan aktifitas pada aplikasi pada semua menu,
2. Cek apakah aktivitas tersebut tercatat di *log server*/aplikasi.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 10].

3.3.10. Pengujian Kerentanan A10: *Server-Side Request Forgery*

Tujuan: Menguji fitur yang melakukan request ke server lain.

Langkah Pengujian:

1. Input *URL* pihak ketiga atau internal (contoh: http://localhost) pada form (contoh: upload foto via *URL*).
2. Observasi *response* aplikasi apakah aplikasi melakukan *request* ke *server* lain atau tidak. Jika aplikasi melakukan *response* sesuai *url* yang diinputkan maka aplikasi memiliki kerentanan.

Untuk tahapan lebih lengkap bisa dilihat pada [lampiran 11].

3.4. Pengaturan Lingkungan Uji

Tahapan ini merupakan tahapan dimana peneliti menyiapkan lingkungan pengujian lokal untuk menguji aplikasi S.E.E.D.S. Dari dokumen uji yang dibuat sebelumnya akan dilakukan persiapan atau installasi aplikasi pada komputer pribadi. Tahapan ini akan memasang aplikasi S.E.E.D.S baik itu bagian *frontend* dan *backend*, alat bantu pengujian aplikasi seperti burp suite, akun palsu untuk pengujian aplikasi. Hal ini dilakukan untuk memudahkan eksekusi skenario pengujian agar bisa dilakukan tanpa hambatan. Sehingga luaran dari tahapan ini adalah aplikasi S.E.E.D.S yang siap diuji.

3.5. Eksekusi Skenario Pengujian

Setelah proses pembuatan kasus uji atau dokumen tahapan-tahapan pengujian keamanan selesai, langkah berikutnya adalah melakukan eksekusi pengujian

sesuai dengan dokumen kasus uji yang telah dibuat sebelumnya. Eksekusi ini dilakukan secara langsung pada aplikasi S.E.E.D.S untuk mengidentifikasi kerentanan dan mengevaluasi tingkat keamanannya. Setiap langkah pengujian dalam kasus uji akan dijalankan secara terstruktur, mulai dari uji kerentanan terhadap ancaman seperti *Injection Attacks* (misalnya, *SQL Injection*), *Broken Authentication*, hingga ancaman *Sensitive Data Exposure*. Untuk setiap jenis kerentanan yang diuji, data pengujian tertentu akan diinputkan ke dalam aplikasi S.E.E.D.S, sementara hasil dan respons aplikasi akan diamati serta dicatat secara rinci. Eksekusi pengujian juga mencakup pemantauan terhadap respons aplikasi dalam menghadapi skenario serangan yang telah dirancang sebelumnya. Misalnya, pada pengujian *Cross-Site Scripting (XSS)*, input skrip berbahaya akan diuji pada fitur-fitur aplikasi seperti formulir pendaftaran, dan pengamatan dilakukan untuk menentukan apakah skrip tersebut berhasil dieksekusi di sisi klien. Demikian pula, untuk *Security Misconfiguration*, pengujian akan dilakukan untuk memastikan tidak ada konfigurasi server atau aplikasi yang membocorkan informasi sensitif.

Hasil dari tahap ini akan memberikan data empiris yang penting, seperti jenis kerentanan yang ditemukan, tingkat keparahan setiap kerentanan, serta area spesifik dalam aplikasi S.E.E.D.S yang memerlukan perbaikan lebih lanjut. Data ini akan menjadi dasar untuk tahap analisis dan rekomendasi selanjutnya dalam penelitian.

3.6. Evaluasi Hasil Pengujian

Tahap ini dilakukan berdasarkan hasil temuan kerentanan yang ada pada aplikasi S.E.E.D.S serta memberikan rekomendasi mitigasi terhadap kerentanan yang ditemukan berdasarkan dokumen OWASP Top 10 serta cara penerapan dari pada mitigasi tersebut. Sebagai contoh, untuk mengatasi serangan injeksi seperti *SQL injection* direkomendasikan untuk memvalidasi input dan menggunakan teknologi *Object-Relational Mapping (ORM).* Pada tahap ini akan memberikan kesimpulan untuk menilai sejauh mana aplikasi S.E.E.D.S telah memenuhi standar keamanan yang ditetapkan dan apakah langkah-langkah mitigasi yang direkomendasikan sudah diterapkan dengan baik. Jika terdapat kerentanan yang ditemukan selama pengujian, kesimpulan ini akan menilai apakah aplikasi telah

mengimplementasikan cara mitigasi yang sesuai berdasarkan pedoman yang ada dalam OWASP Top 10.

BAB 4 HASIL PERCOBAAN DAN ANALISIS

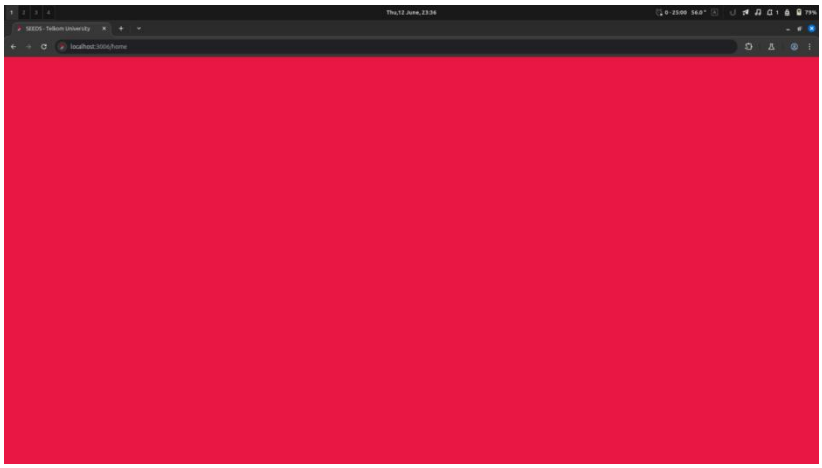
4.1. Hasil Percobaan

4.1.1. Hasil Uji Kerentanan A01: *Broken Access Control*

Tabel 4.1 Tabel Hasil Pengujian A01: *Broken Authentication*

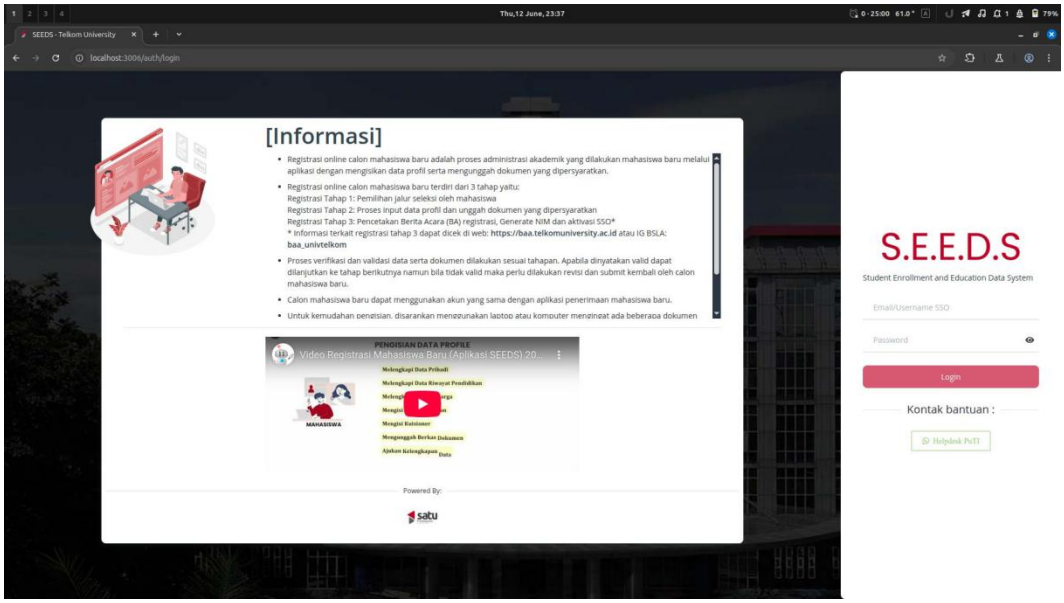
Skenario	Hasil
Mengakses halaman yang membutuhkan autentikasi tanpa melakukan autentikasi	Tidak ditemukan kerentanan
Manipulasi aplikasi dengan memalsukan token	Tidak ditemukan kerentanan
Modifikasi data pengguna orang lain	Ditemukan keretanan
Manipulasi hak akses pada token autentikasi	Tidak ditemukan kerentanan

Hasil skenario mengakses halaman yang membutuhkan autentikasi diakses tanpa melakukan autentikasi hasilnya adalah aplikasi melalakukan redirect ke halaman login kembali.



Gambar 4.1 Loading Aplikasi S.E.E.D.S

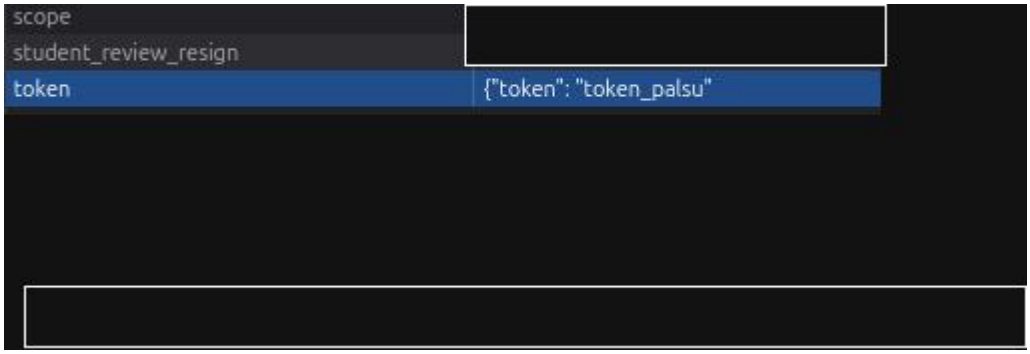
Gambar diatas adalah halaman loading ketika mencoba akses halaman dashboard dimana halaman tersebut memerlukan autentikasi, lalu aplikasi melakukan redirect ke halaman login sebagai bentuk pencegahan kepada user yang tidak melakukan autentikasi.



Gambar 4.2 Halaman Login Aplikasi S.E.E.D.S

Maka pada skenario ini aplikasi tidak ditemukan kerentanan. Pada skenario memanipulasi aplikasi dengan memalsukan token supaya menyerupai pengguna yang telah melakukan autentikasi. Pada aplikasi didapatkan bahwa setelah melakukan autentikasi, maka aplikasi akan menyimpan token autentikasi di localstorage, dengan mencoba memalsukan token sembarang untuk mengetahui apakah aplikasi melakukan validasi terhadap token tersebut atau tidak ketika pengguna belum melakukan autentukasi, maka pada localstorage bagian token akan diisi token palsu.

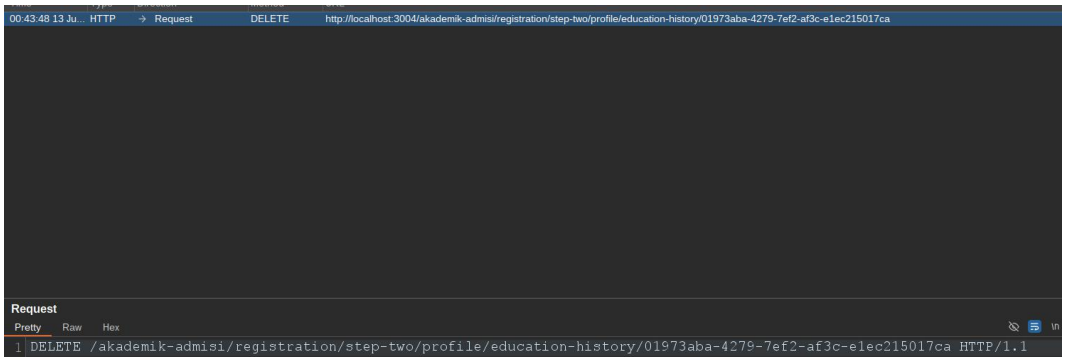
Gambar 4.3 Isi Localstorage Aplikasi S.E.E.D.S



Gambar 4.4 Pengisian Token Palsu Pada Localstorage Aplikasi

Setelah diisi token palsu dan mencoba kembali mengakses halaman dashboard, aplikasi melakukan redirect kembali ke halaman login sama seperti skenario pertama dan seperti pada gambar 4.1. Maka pengujian skenario ini lolos dan tidak ditemukan kerentanan.

Lalu pada skenario memodifikasi data milik orang lain, ternyata ditemukan kerentanan yaitu berhasil melakukan menghapus data milik orang lain contohnya pada data riwayat pendidikan. Dengan menggunakan *burp suite* bisa dilihat bagaimana aplikasi melakukan *request* ke server.

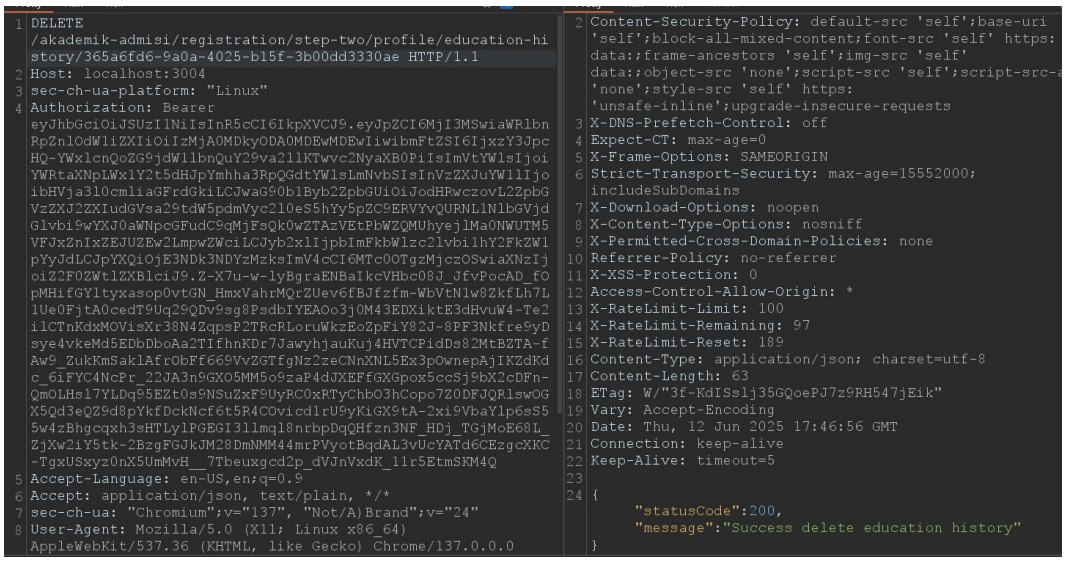


Gambar 4.5 Intercept Request Pada Aplikasi Burp Suite

Lalu pada *burp suite request* di simpan ke repeter untuk di ganti id dari riwayat pendidikan menjadi pengguna lain.

Gambar 4.6 Mengganti ID Data Pada Burp Suite

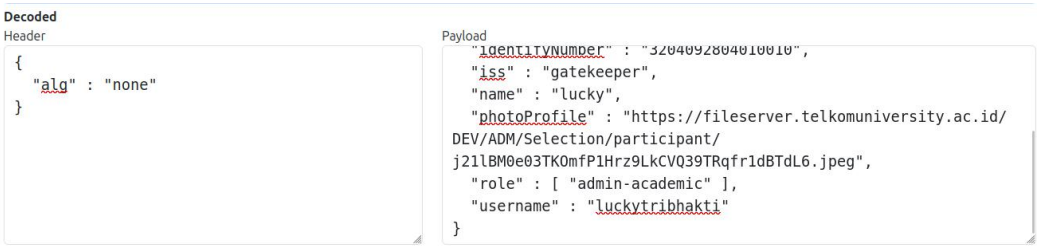
Ternyata server pun meresponse dengan sukses dan ketika di cek data pada *database* pun, data milik pengguna orang lain pun sudah tidak ada.



Gambar 4.7 Hasil Repeater Request Ke Aplikasi S.E.E.D.S

Bisa dilihat pada gambar diatas, dilakukan penghapusan data dengan id 365a6fd6-9a0a-4025-b15f-3b00dd3330ae dan *response* aplikasi menyatakan sukses. Maka pada kategori ini aplikasi memiliki kerentanan.

Selanjutnya pada skenario terakhir yaitu manipulasi hak akses pada token autentikasi, dimana token autentikasi akan diubah hak aksesnya mejadi *role admin* pada aplikasi. Dengan menggunakan bantuan aplikasi *webwolf* dari OWASP sebagai alat untuk mengubah token jwt dan mengubah rolenya seperti ini.



Gambar 4.8 Mengganti Payload JWT Menggunakan Webwolf

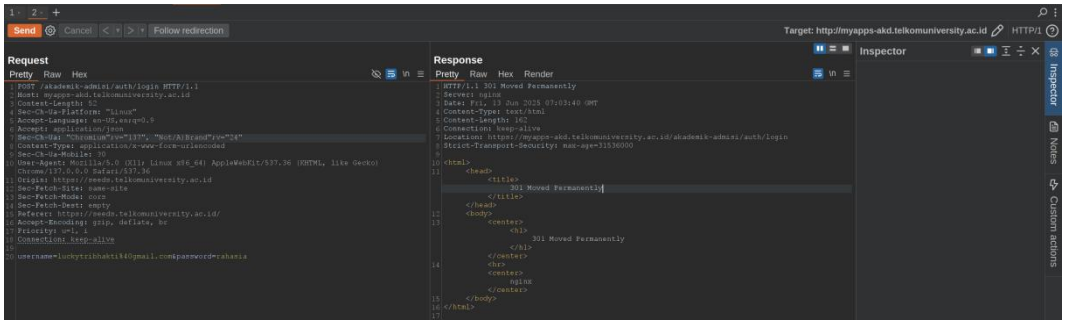
Dan didapatkan token jwt yang sudah jadi sehingga token tersebut hanya perlu dimasukan ke *localstorage* seperti skenario kedua yaitu memanipulasi aplikasi dengan memalsukan token. Hasilnya aplikasi melakukan redirect ke halaman login karena aplikasi menyadari bahwa token tidak valid. Pada kategori A01: *Broken Access Control* ini maka didapatkan 1 skenario yang berhasil ditemukan kerentanan yaitu memodifikasi data milik orang lain.

4.1.2. Hasil Uji Kerentanan A02: *Cryptographic Failure*

Tabel 4.2 Tabel Hasil Pengujian A02: *Cryptographic Failure*

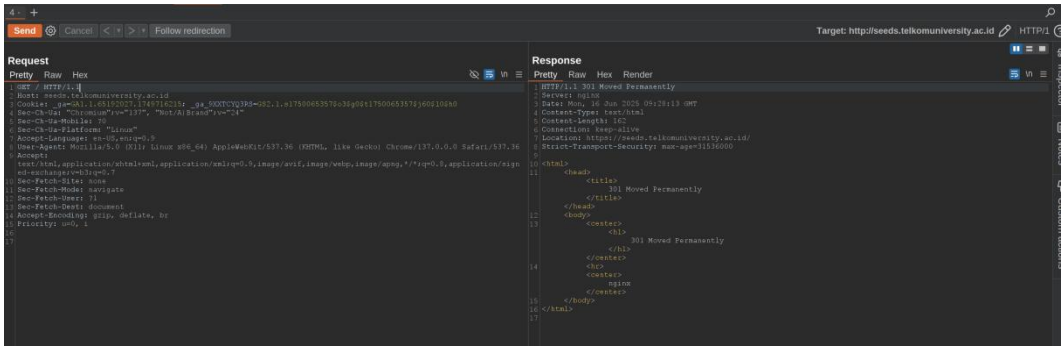
Skenario	Hasil
Https downgrade	Tidak ditemukan kerentanan
Enkripsi sensitif data	Ditemukan kerentanan

Pada hasil skenario *https downgrade* ditemukan bahwa aplikasi melakukan *redirect* secara otomatis langsung ke https. Hasil *redirect* tersebut dapat dilihat pada *burp suite* dimana *server meresponse 301* atau *redirection*.



Gambar 4.9 Request Aplikasi Menggunakan Burp Suite

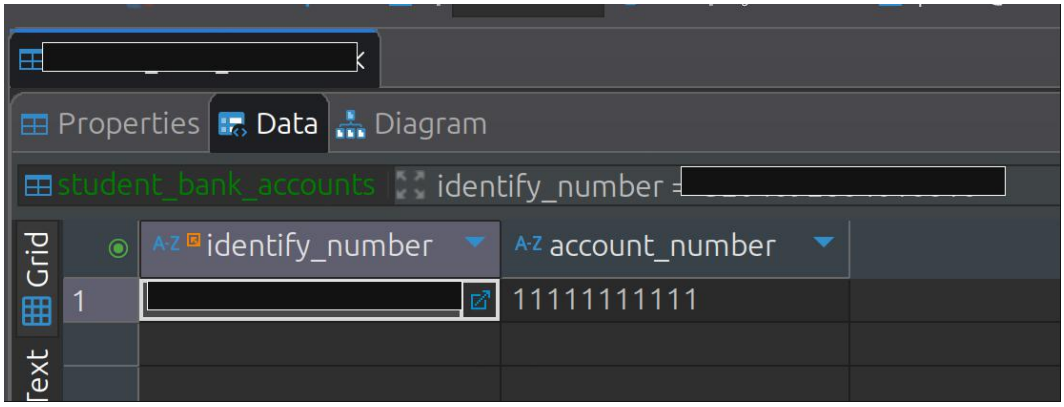
Seperti pada gambar diatas, dilakukan *request* pada aplikasi dibagian sisi *backend* dan aplikasi melakukan *response 301* atau *redirection*, begitu pula dengan aplikasi pada sisi *frontend.*



Gambar 4.10 Hasil Request Ke Aplikasi S.E.E.D.S Menggunakan Burp Suite

Dapat dilihat dari *burp suite request* dilakukan menggunakan koneksi http dan aplikasi meresponse dengan *status code 301* atau *redirection* maka hal tersebut dapat dikatakan bahwa skenario ini tidak ditemukan kerentanan.

Lalu pada skenario kedua yaitu enkripsi data sensitif setelah memasukan data pribadi pada aplikasi, salah satunya yaitu data rekening pribadi didapatkan pada database bahwa data rekening disimpan dalam bentuk *plain text* artinya, pada skenario ini dapat ditemukan kerentanan.



Gambar 4.11 Hasil Penyimpanan Data No Rekening

4.1.3. Hasil Uji Kerentanan A03: *Injection*

Tabel 4.3 Tabel Hasil Pengujian A03: *Injection*

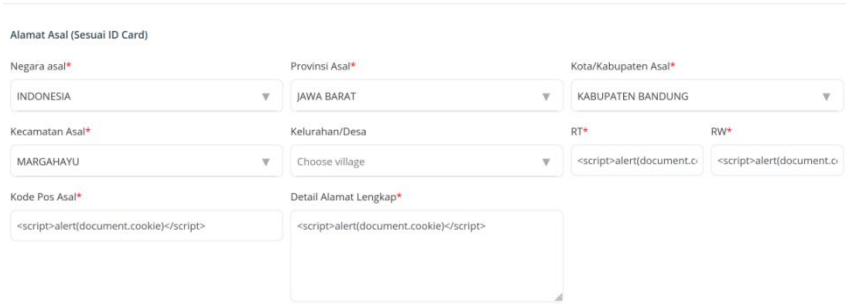
Skenario	Hasil
SQL injection	Tidak ditemukan kerentanan
XSS	Tidak ditemukan kerentanan

Hasil dari skenario *SQL injection* didapatkan serangan *SQL injection* tidak berhasil di eksekusi, artinya aplikasi memperlakukan kode *SQL* pada input sebagai teks biasa bukan sebagai kode *SQL* yang harus di eksekusi. Contoh dari serangan *SQL injection* sebagai berikut

Gambar 4.12 Percobaan *SQL Injection*

Dilakukan proses *SQL injection* pada input data pribadi pada nama lengkap dan melakukan submit hasilnya tabel '*test*' tidak terhapus dan masih ada.

Lalu pada skenario ke 2 yaitu *XSS* hasilnya pun sama seperti *SQL injection* yaitu kode *javascript* tidak tereksekusi pada aplikasi atau browser artinya aplikasi sama memperlakukan kode inputan *javascript* seperti teks biasa. Berikut contohnya



Gambar 4.13 Percobaan *XSS*

Pada inputan alamat asal diinputkan kode *javascript* untuk mengambil *cookies* pada browser. Hal tersebut aplikasi memperlakukan nya sebagai teks biasa berikut hasilnya

Gambar 4.14 Hasil XSS Pada Aplikasi S.E.E.D.S

Setelah dilakukan *XSS* aplikasi tidak mengeksekusi kode *javascript* tersebut.

4.1.4. Hasil Uji Kerentanan A04: *Insecure Design*

Untuk hasil uji pada point ini, didapatkan pada pengujian dari A01-A10 adalah sebagai berikut:

Tabel 4.4 Tabel Hasil Pengujian A04: *Insecure Design*

Kategori	Ditemukan Kerentanan atau Tidak
A01: *Broken Access Control*	Pada skenario ini ditemukan kerentanan
A02: *Cryptographic Failure*	Pada skenario ini ditemukan kerentanan
A03: *Injection*	Pada skenario ini tidak ditemukan kerentanan
A05: *Security Misconfiguration*	Pada skenario ini ditemukan kerentanan
A06: *Vulnerable and Outdated Components*	Pada skenario ini ditemukan kerentanan
A07: *Identification and Authentication Failures*	Pada skenario ini ditemukan kerentanan
A08: *Software and Data Integrity Failures*	Pada skenario ini tidak ditemukan kerentanan
A09: *Security Logging and Monitoring*	Pada skenario ini ditemukan kerentanan

Failures	
A10: *Server-Side Request Forgery (SSRF)*	Pada skenario ini tidak ditemukan kerentanan

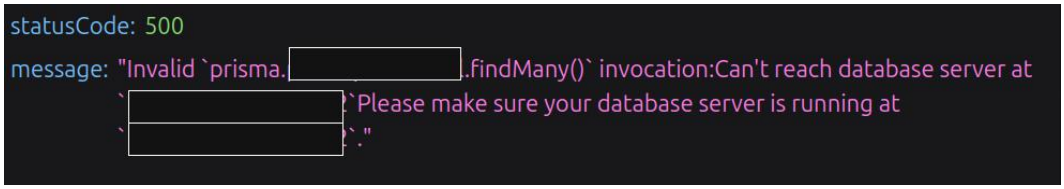
Hasil dari pengujian, terdapat beberapa kategori OWASP yang diuji ditemukan beberapa kategori OWASP yang masih tidak lulus uji atau ditemukannya kerentanan berdasarkan skenario yang telah dilakukan. Untuk menentukan apakah aplikasi lulus uji atau lulus kategori dari A04: Insecure Design adalah dengan melihat dokumen perancangan aplikasi S.E.E.D.S sendiri. Ternyata hasilnya tidak ditemukannya dokumen perancangan mengenai bagaimana aplikasi akan di bangun. Sehingga dengan adanya ini, pada kategori ini dinyatakan A04: Insecure Design aplikasi tidak lolos atau memiliki kerentanan pada kategori ini.

4.1.5. Hasil Uji Kerentanan A05: *Security Misconfiguration*

Tabel 4.5 Tabel Hasil Pengujian A05: *Security Misconfiguration*

Skenario	Hasil
Eksposur informasi sistem melalui error detail	Ditemukan kerentanan
CSRF	Tidak ditemukan kerentanan

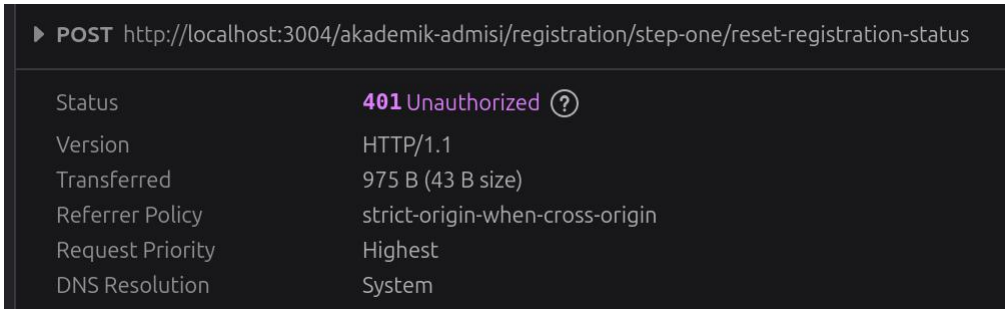
Pada skenario eksposur informasi sistem melalui *error detail* ditemukan pesan *error* yang menampilkan data ip dari *database* beserta *port database*, serta menampilkan fungsi pemanggilan ke *database* menggunakan *ORM prisma.*



Gambar 4.15 Hasil *Response Error* Pada Aplikasi S.E.E.D.S

Ini sangat berbahaya karena penyerang mendapatkan informasi *ip database* beserta portnya dan mendapatkan informasi bahwa aplikasi di bangun menggunakan *ORM prisma* sehingga penyerang bisa memanfaatkan kerentanan yang ada pada library prisma untuk menyerang aplikasi.

Lalu pada skenario CSRF, aplikasi tidak bisa di tembus melalui serangan CSRF.



Gambar 4.16 Hasil serangan *CSRF* Pada Aplikasi S.E.E.D.S

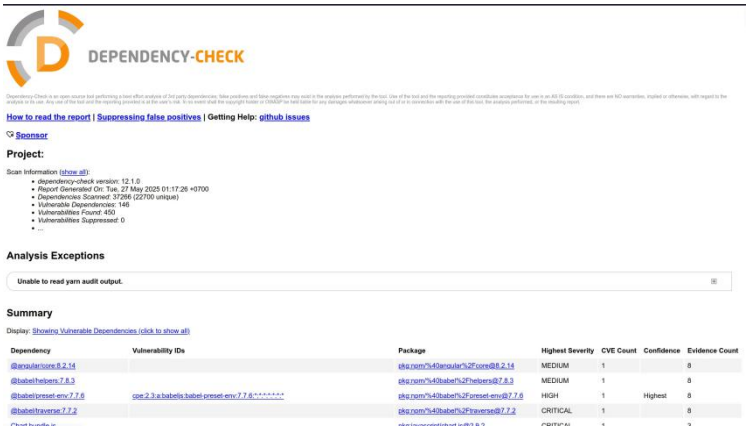
Request kepada aplikasi tidak bisa ditembus melalui aplikasi lain, dalam hal ini adalah link penipuan penyerang. Sehingga aplikasi aman tidak rentang dalam serangan CSRF namun untuk skenario exposur informasi sistem melalui *error detail* terjadi sehingga aplikasi rentan dalam kategori A05: *Security Misconfiguration.*

4.1.6. Hasil Uji Kerentanan A06: *Vulnerable and Outdated Components*

Tabel 4.6 Tabel Hasil Pengujian A06: *Vulnerable and Outdated Components*

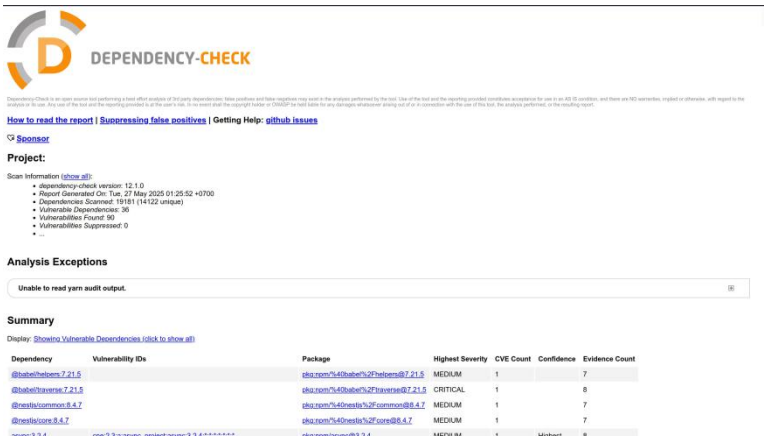
Aplikasi	Memiliki library yang vurnerable dan outdated
Frontend	Ya
Backend	Ya

Berdasarkan hasil uji, ditemukan bahwa pada aplikasi bagian *frontend* dan *backend* ditemukan adanya *library* yang sudah *outdated* atau usang dan memiliki kerentanan pada *library* tersebut.



Gambar 4.17 Hasil Cek *Depedency* Pada Aplikasi S.E.E.D.S Bagian *Frontend*

Pada bagian *frontend* terdapat 146 *library* yang memiliki kerentanan dan pada 146 *library* tersebut ditemukan 450 kerentanan. Contoh pada gambar diatas pada *library* @angular/core:8.2.14 terdapat 1 kerentanan dan tingkat kerentanannya medium, sehingga ini perlu diwaspadai.



Gambar 4.18 Hasil Cek *Depedency* Pada Aplikasi S.E.E.D.S Bagian *Backend*

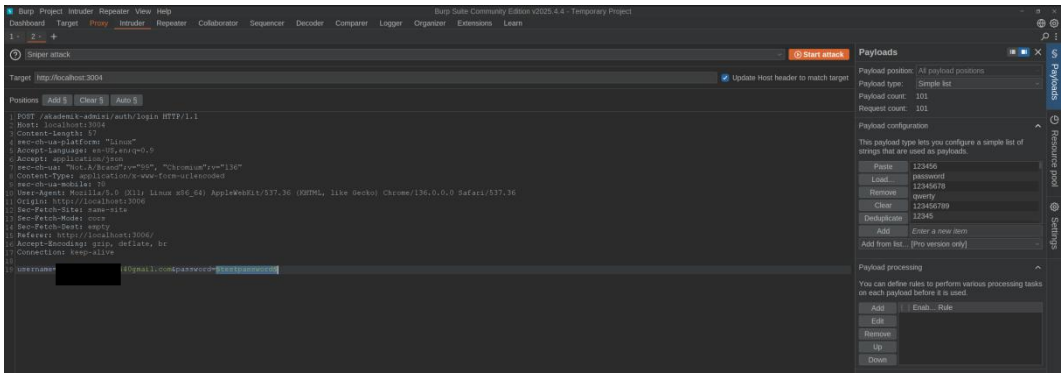
Pada bagian aplikasi *backend,* ditemukan *library* yang memiliki kerentanan sebanyak 36 dan pada *library* tersebut ditemukan 90 kerentanan. Contoh pada gambar diatas *library* @nestjs.core:8.4.7 memiliki kerentanan sebanyak 1 dan tingkat kerentanannya adalah medium, ini pun perlu diwaspadai agar aplikasi aman dari serangan penyerang. Karena bisa saja penyerang memanfaatkan kelemahan pada library ini untuk menyerang aplikasi.

4.1.7. Hasil Uji Kerentanan A07: *Identification and Authentication Failures*

Tabel 4.7 Tabel Hasil Pengujian A07: *Identification and Authentication Failures*

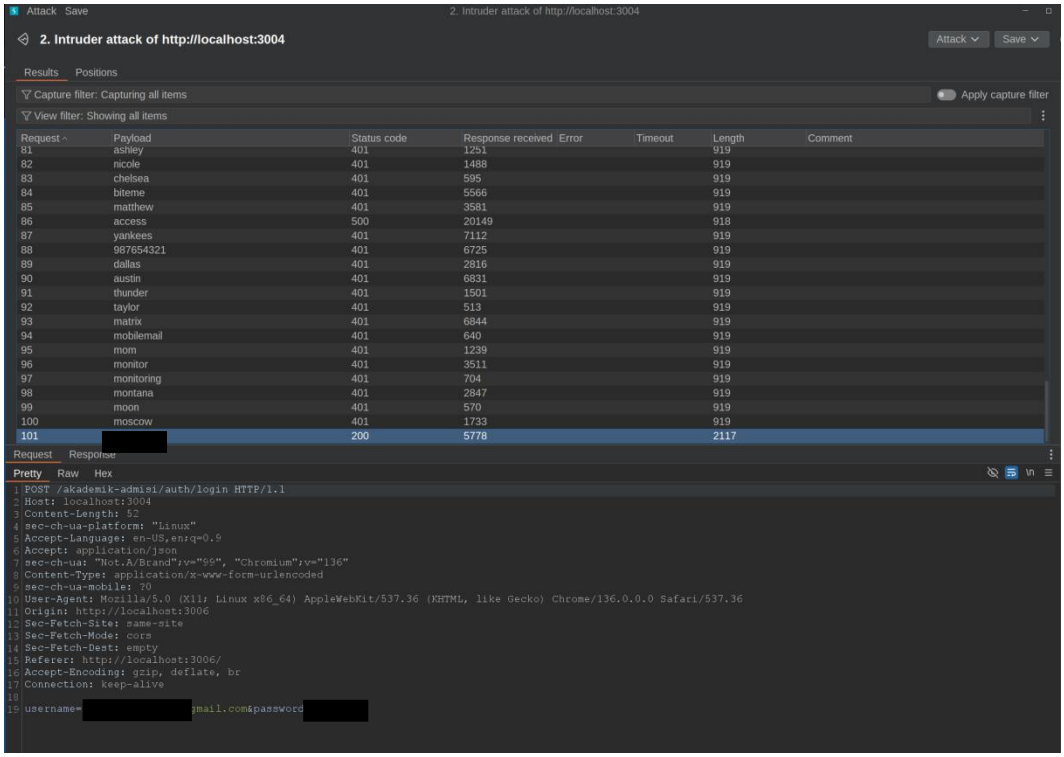
Percobaan	Hasil
Brute Force	Ditemukan kerentanan
Skenario autentikasi pada komputer publik	Ditemukan kerentanan (Selama 1 jam masih bisa diakses)

Pada skenario *brute force login*, aplikasi berhasil di bobol dengan cara *brute force.* Percobaan ini dilakukan dengan skenario bahwa penyerang mengetahui *username/email* pengguna dan melakukan *brute force password* ke aplikasi. Aplikasi tidak melakukan *response status 429 (To Many Requests)*, ini sangat diperlukan karena dengan adanya *rate limiting* pada aplikasi jika percobaan *request* melebihi batas yang telah ditentukan maka akan ada *cooldown* atau waktu untuk bisa melakukan *request* kembali untuk melakukan *login*, sehingga akan memperlambat proses penyerangan dan penyerang akan lebih lama bisa membobol suatu akun.



Gambar 4.19 Proses Serangan *Brute Force* Menggunakan Burp Suite

Percobaan dilakukan dengan menggunakan *burp suite* dan dicoba dengan 100 kali *request* dan *request* ke 101 adalah *password* asli dan *request* dilakukan ke *endpoint /akademik-admisi/auth/login.*



Gambar 4.20 Hasil Serangan *Brute Force* Pada Aplikasi S.E.E.D.S

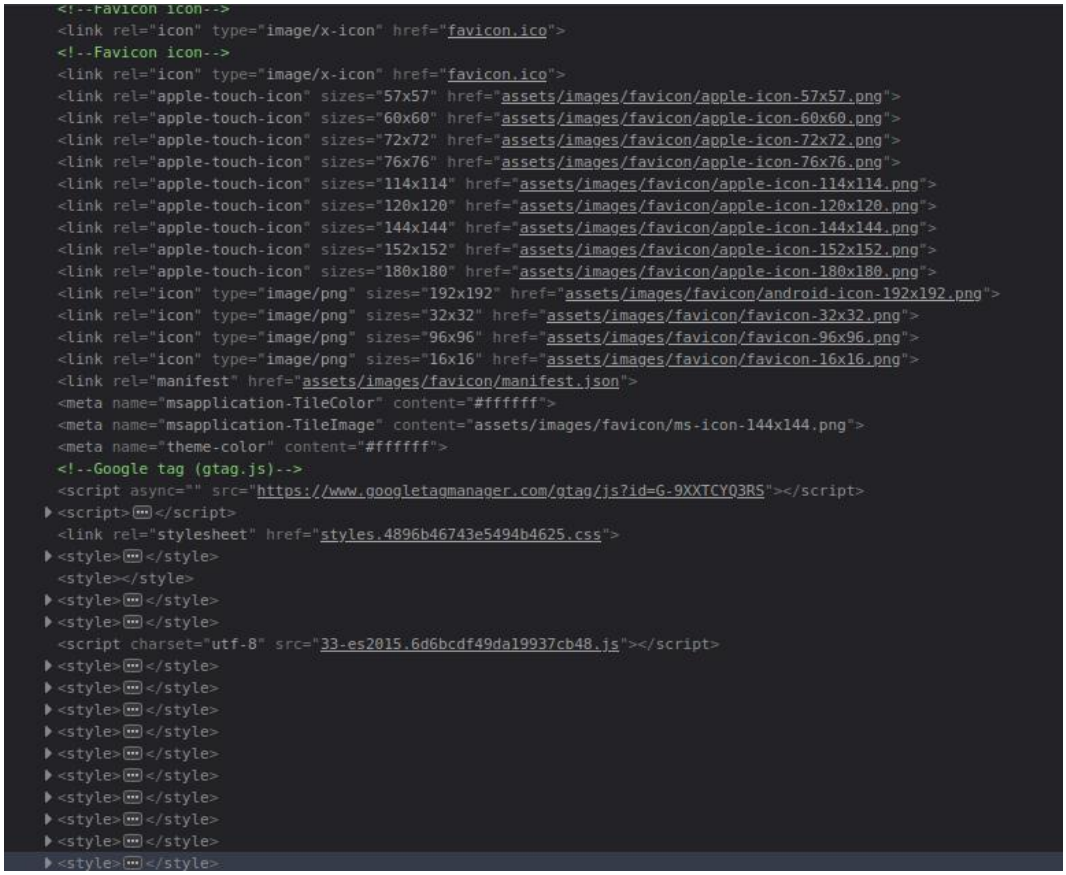
Dari gambar diatas dilakukan 101 request dimana *request* ke 101 berhasil dan dari dari 100 *request* sebelumya tidak ada *response* yang menunjukan status 429. Rate limiting penting karena untuk mencegah *brute force attack* sehingga memperlambat proses peretasan akun.

Lalu pada skenario penggunaan komputer publik setelah 1 jam saat aplikasi di tutup melalui *close browser*, aplikasi tetap bisa diakses dengan status tetap terautentikasi atau terlogin. Ini berbahaya jika pengguna mengakses aplikasi daripada komputer publik seperti dari warung internet ataupun komputer perpustakaan. Aplikasi akan terlogout sendiri jika sudah 1x24 jam.

4.1.8. Hasil Uji Kerentanan A08: *Software and Data Integrity Failures*

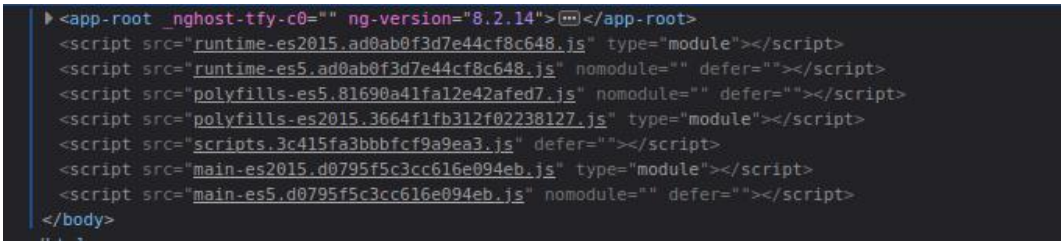
Tabel 4.8 Tabel Hasil Pengujian A08: *Software and Data Integrity Failures*

Aplikasi	Library berasal dari sumber yang terpercaya
Frontend	Ya
Backend	Ya



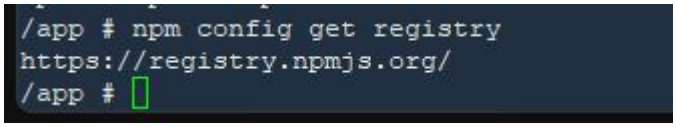
Gambar 4.21 Hasil Cek *Library CSS* Pada Aplikasi S.E.E.D.S

Pada aplikasi bagian *frontend* tidak ditemukan *library* css pihak ketiga yang berasal dari CDN. Jika ada, maka *library* harus berasal dari sumber yang terpercaya seperti https://cdnjs.com/ atau https://www.jsdelivr.com/ pada kasus ini tidak ditemukan *library* yang berasal dari sumber tersebut.



Gambar 4.22 Hasil Cek *Library javascript* Pada Aplikasi S.E.E.D.S

Begitupun pada bagian *javascript*, aplikasi tidak ditemukan memiliki *library* pihak ketidak menggunakan CDN, dan pada bagian aplikasi *backend* aplikasi mengunduh *library* dari https://registry.npmjs.org merupakan sumber asli untuk mengunduh *library* nodejs.



Gambar 4.23 Hasil Cek sumber Library Pada Aplikasi S.E.E.D.S Backend

Pada gambar diatas merupakan hasil uji untuk mengetahui apakah aplikasi mengunduh dari sumber terpercaya atau tidak.

4.1.9. Hasil Uji Kerentanan A09: *Security Logging and Monitoring Failures*

Tabel 4.9 Tabel Hasil Pengujian A09: *Security Logging and Monitoring Failures*

Menu	Ditemukan Log	Isi Log Sensitif
Login	Ya	Ya
Registrasi Tahap 1	Ya	Ya
Registrasi Tahap 2	Ya	Ya
Registrasi Tahap 3	Ya	Ya
Undur Diri	Ya	Ya

1. Login

Gambar 4.24 Hasil Log Proses *Login* Aplikasi S.E.E.D.S

Pada menu pertama yaitu *login* aplikasi sudah menggunakan log, sehingga aksi *user* sudah tercatat dan ini merupakan langkah yang bagus karena log ini digunakan sebagai pengawasan, keamanan dan pemeliharaan aplikasi. Namun pada aksi *login*, aplikasi melakukan *login* data kredentials dengan *password* pengguna juga, sehingga ini salah dan berbahaya dikarenakan jika seorang *hacker* bisa membobol server aplikasi, maka sudah jelas *hacker* sudah mendapatkan kredentials yang lengkap.

2. Registrasi Tahap 1

Gambar 4.25 Hasil Log Proses Registrasi Tahap 1 Aplikasi S.E.E.D.S

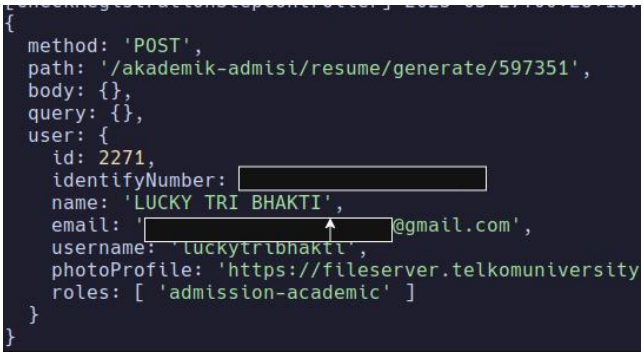
Pada menu kedua yaitu registrasi tahap 1, ditemukan juga log aksinya dan ini sama seperti pada halaman login karena melakukan logging dan memiliki data yang sifatnya sensitif yaitu pada data *"identifyNumber"* atau NIK pengguna. Data NIK menurut UU no 27 tahun 2022 tentang perlindungan data pribadi, data NIK merupakan data yang perlu dijaga sesuai pasal 4 ayat 3f yaitu "Data pribadi yang dikombinasikan untuk mengidentifikasi seseorang". Sehingga pada nenu registrasi tahap 1 ini sudah betul melakukan log tetapi log memuat sensitif [24]

3. Registrasi Tahap 2

Gambar 4.26 Hasil Log Proses Registrasi Tahap 2 Aplikasi S.E.E.D.S

Pada menu kedua yaitu registrasi tahap 2 ditemukan juga log aksinya. Sama seperti halaman registrasi tahap 1 dimana pada tahap 2 ditemukan log dengan detail bank pengguna yaitu *"accountName"* dan *"accountNumber"* dimana ini merupakan nama pengguna *bank* beserta nomor rekening *bank* pengguna.

4. Registrasi Tahap 3



Gambar 4.27 Hasil Log Proses Registrasi Tahap 3 Aplikasi S.E.E.D.S

Pada menu kedua yaitu registrasi tahap 3 ditemukan juga log aksinya. Sama seperti halaman registrasi tahap 1 dan 2 dimana pada tahap 3 ditemukan log pada data *"identifyNumber"* atau NIK pengguna. Dimana

data NIK ini adalah data sensitif dan perlu dijaga sesuai peraturan UU no 27 tahun 2022 tentang perlindungan data pribadi [sitasi]

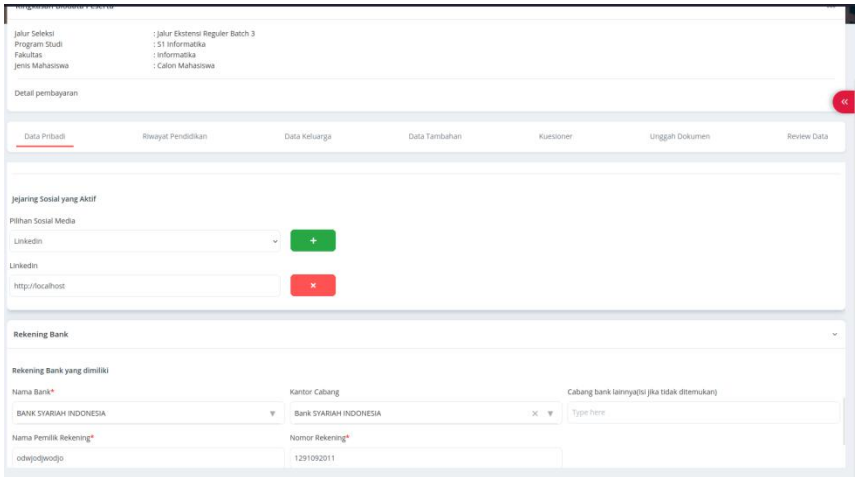
5. Undur Diri

Gambar 4.28 Hasil Log Proses Undur Diri Aplikasi S.E.E.D.S

Pada halaman undur diripun sama seperti sebelumnya, dimana terdapat log namun tetap melakukan log data sensitif seperti NIK, nama bank dan nomor rekening pengguna.

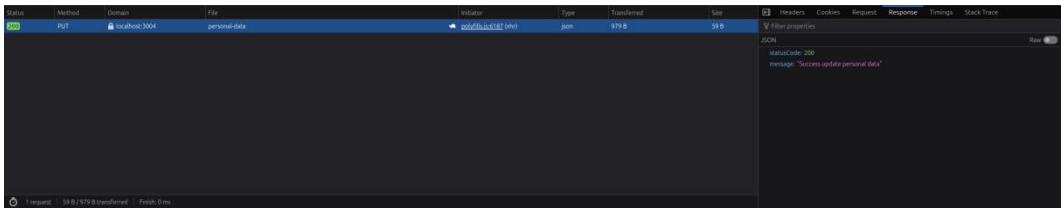
4.1.10. Hasil Uji Kerentanan A10: *Server-Side Request Forgery*

Hasil pada pengujian A10 dinyatakan tidak ditemukan kerentanan. Karena pada aplikasi tidak ditemukan form input yang bisa melakukan *request* pada aplikasi lain atau *resource* lain.



Gambar 4.29 Proses Input Pada Form Sosial Media

Pada menu registrasi tahap 2, ditemukan *form input* untuk memasukan sosial media, dimana pada *input* ini menggunakan tipe bebas sehingga pengguna bisa memasukan *username* sosial medianya saja ataupun url menuju sosial media. Pada percobaan ini ditest dengan memasukan url http://localhost dengan tujuan aplikasi akan *request* ke url tersebut dan harapan aplikasi akan meresponse atau menampilkan halaman *webserver* dari aplikasi tersebut.



Gambar 4.30 Response Aplikasi S.E.E.D.S

Namun aplikasi hanya meng*update* data sosial media pengguna dan tidak melakukan *request* ke url yang dituju, sehingga aplikasi pada uji A10: *Server-side request forgery* ini dinyatakan tidak ditemukan kerentanan. Bisa dilihat aplikasi hanya meresponse bahwa request sukses dibuat. sehingga kerentanan A10 ini tidak terjadi pada aplikasi ini.

4.2. Analisis

4.2.1. Analisis Uji Kerentanan A01: *Broken Access Control*

Sesuai dengan hasil uji bahwa pada skenario memodifikasi data pengguna lain pada aplikasi gagal lolos pengujian. Hal tersebut menjadikan aplikasi masih dapat memodifikasi data milik pengguna orang lain, pada kasus ini adalah menghapus data riwayat pendidikan. Hal Ini menyebabkan hasil yang berbahaya karena pengguna bisa menghapus riwayat pendidikan milik orang lain sesuka hati, walaupun tidak bisa melalui aplikasi pada bagian *frontend* secara langsung, tapi pengguna tetap bisa melakukan aksi melalui aplikasi bagian *backend.* Hal ini bisa terjadi karena aplikasi tidak memvalidasi apakah pengguna melakukan akses ke sumber yang valid atau tidak. Contohnya pada kasus ini pengguna bisa menghapus daftar riwayat pendidikan milik pengguna lain karena aplikasi tidak

memvalidasi apakah *id* dari riwayat pendidikan tersebut di akses oleh pengguna yang sah.

4.2.2. Analisis Uji Kerentanan A02: *Cryptographic Failure*

Berdasarkan hasil uji pada skenario enkripsi sensitif data, ditemukan bahwa aplikasi tidak melakukan enkripsi pada data rekening bank. Hal ini bisa terjadi karena saat proses *input* ke *database* aplikasi tidak melakukan enkripsi data rekening bank data tersebut. Sehingga pada *database* data rekening bank pengguna tetaplah merupakan teks yang asli tidak berupa teks acak hasil enkripsi dari teks asli.

4.2.3. Analisis Uji Kerentanan A03: *Injection*

Pada hasil uji didapatkan semua skenario berhasil lolos uji atau aplikasi tidak dapat dimasukan kode berbahaya. Dimana ini merupakan hasil yang baik karena aplikasi memastikan input pengguna untuk di validasi dan dilakukan sanitasi supaya input masuk kedalam database secara aman.

4.2.4. Analisis Uji Kerentanan A04: *Insecure Design*

Berdasarkan dokumen perancangan aplikasi S.E.E.D.S tidak ditemukan dokumen perancangan mengenai keamanan aplikasi, contohnya pada logging aplikasi tidak ditemukan dokumen mengenai bagaimana dan data apa saja yang harus di logging pada aplikasi atau contoh lainnya mengenai *Vurnerable and Outdated Components* tidak terdapat perancangan harus seperti apa dan bagaimana mekanismenya untuk mengupdate *library* pihak ketiga. Sehingga aplikasi S.E.E.D.S masuk ke dalam kerentanan A04: *Insecure Design.* Menurut OWASP untuk mencegah hal ini terjadi yaitu dengan seharusnya hal ini dapat dilakukan pada sebelum pembuatan aplikasi dilakukan sehingga dapat menerapkan alur pengembangan yang aman oleh profesional untuk mendesain keamanan aplikasi[28].

4.2.5. Analisis Uji Kerentanan A05: *Security Misconfiguration*

Berdasarkan hasil uji pada skenario *exposur* informasi sistem melalui error detail ini disebabkan karena kurangnya penanganan *error* pada aplikasi sehingga menyebabkan pesan *error* yang terlalu detail sampai kepada pengguna [29] . Hal

ini bisa terjadi karena biasanya pengembang lupa mengatur konfigurasi pada file .env pada variabel ENV dimana nilai dari variabel ENV=*development* sehingga aplikasi menampilkan detail *error* yang lengkap untuk informasi ke pengembang aplikasi bahwa ada *error*.

4.2.6. Analisis Uji Kerentanan A06: *Vurnerable and Outdated Components*

Dari hasil pengujian ditemukan pada aplikasi bagian *frontend* maupun *backend* ditemukan *library* yang memiliki kerentanan. Ini biasa terjadi jika pengembang aplikasi lupa melakukan pembaharuan secara rutin untuk pustaka pihak ke-3 yang digunakannya. Pembaharuan pada pustaka pihak ke-3 penting dilakukan karena kerentanan aplikasi bukan hanya bisa diserang melalui aplikasi itu sendiri melainkan melalui kerentanan pustaka pihak ke-3 yang digunakan aplikasi.

4.2.7. Analisis Uji Kerentanan A07: *Identification and Authentication Failures*

Pada hasil uji skenario *brute force* login aplikasi tidak menolak *request* jika itu terjadi dan pada skenario *login* pada komputer publik aplikasi tidak melakukan *logout* secara otomatis. Pada *brute force* aplikasi tidak tahu permintaan mana yang termasuk *brute force* atau tidak sehingga seharusnya aplikasi menerapkan *rate limiting* untuk mencegah 1 *ip* melakukan permintaan secara terus menerus dalam rentang waktu tertentu dan skenario komputer publik aplikasi pun tidak mengetahui apakah pengguna sedang *idle* tidak melakukan apa-apa atau tidak seharusnya aplikasi mengetahui pengguna sedang *idle* atau tidak untuk melakukan *logout* secara otomatis.

4.2.8. Analisis Uji Kerentanan A08: *Software and Data Integrity Failures*

Hasil pengujian poin ini menunjukan bahwa aplikasi memang memiliki *library* pihak ke 3 dalam pengembangannya, namun *library* di*install* melalui sumber yang terpercaya. Pada aplikasi bagian *backend* contohnya, *library* di*install* melalui https://registry.npmjs.org/ hal tersebut artinya *npmjs* ini adalah *website official* dari *nodejs* sebagai wadah para pengembang dalam meng*install library*, sehingga sudah dipastikan keamanannya. Pada sisi *frontend* tidak ditemukannya *library* pihak ketiga yang melalui *CDN*, sehingga aplikasi tidak menggunakan sumber kode dari luar yang berbahaya jadi aplikasi tidak memiliki kerentanan pada kategori ini.

4.2.9. Analisis Uji Kerentanan A09: *Security Logging and Monitoring Failures*

Berdasarkan hasil uji menunjukan bahwa aplikasi sudah tepat karena memiliki *logging* sebagai *history* dari aksi yang dilakukan pengguna karena memang *log* diperuntukan untuk developer dalam memelihara aplikasi apabila terjadi suatu kesalahan [32]. Namun *log* tidak seharusnya menampilkan data yang sifatnya sensitif seperti *password login* atau akun bank [32] karena untuk mengantisipasi jika server dibobol oleh penyerang dan penyerang melihat log aplikasi, maka penyerang dengan mudah mendapatkan identitas atau data pribadi seseorang. Oleh karena itu maka aplikasi tidak lolos dari kerentanan A09: *Security Logging and Monitoring Failures.* Hal ini bisa terjadi bisa saja karena pengembang lupa menghapus *log* data sensitif atau bisa saja aplikasi tidak di desain sejak awal bagaimana cara melakukan *log* data apa saja yang harus dimunculkan pada *log.*

4.2.10. Analisis Uji Kerentanan A10: *Server-Side Request Forgery*

Hasil uji menunjukan bahwa aplikasi aman dari kerentanan A10: *Server-Side Request Forgery.* Dikarenakan pada aplikasi memang hanya mencatat data dari pada sosial media pengguna saja, tidak sampai aplikasi melakukan *request* ke *url* sosial media tersebut dan dilakukan *scraping* untuk mendapatkan data dari pengguna, jadi hanya sebatas pencatatan sosial media dari pengguna saja. Karena memang kerentanan A10: *Server-Side Request Forgery* ini hanya terjadi jika aplikasi melakukan *reques*t ke inputan *url* pengguna.

4.3. Rekomendasi Mitigasi Kerentanan Pada Aplikasi S.E.E.D.S

4.3.1. Rekomendasi Mitigasi Kerentanan A01: *Broken Access Control*

Untuk memitigasi hal ini terjadi, aplikasi harus dilakukan pembaharuan yang dimana harus melakukan pengecekan akses kontrol terhadap data atau objek yang diakses contoh dalam kasus ini adalah *id* dari riwayat pendidikan pengguna ketika ada *request* ke aplikasi maka aplikasi harus memastikan akses kontrol *id* tersebut apakah benar dari pengguna yang sah atau tidak [25] . Lalu pengaturan akses kontrol ini hanya bisa efektif pada aplikasi *backend* bukan pada *frontend*, sehingga pengecekan akses kontrol, apakah pengguna bisa melakukan sesuatu yang sah atau tidak pun harus di lakukan pada *backen*d bukan pada *frontend* [26].

4.3.2. Rekomendasi Mitigasi Kerentanan A02: *Cryptographic Failures*

Untuk memitigasi hal ini aplikasi perlu di update untuk menerapkan enkripsi yang sifatnya sensitif jika data disimpan pada *database* sesuai apa yang dikatakan OWASP [27]. Lalu untuk enkripsi oleh OWASP dikatakan bahwa enkripsi tidak boleh menggunakan fungsi enkripsi yang sudah usang seperti *MD5, SHA1, PKCS number 1 v1.5* [27] Hal ini pun sesuai dengan undang-undang nomor 27 tahun 2022 bab 3 pasal 4 ayat 2 point F yaitu data keuangan pribadi merupakan hal yang harus dijaga dan sesuai OWASP pun data yang sifatnya sensitif harus dilakukan enkripsi terlebih dahulu[27][24].

4.3.3. Rekomendasi Mitigasi Kerentanan A03: *Injection*

Tidak ada yang harus di mitigasi dari ketegori ini karena pada aplikasi S.E.E.D.S untuk kategori ini berhasil lulus tes atau tidak ditemukan kerentanan.

4.3.4. Rekomendasi Mitigasi Kerentanan A04: *Insecure Design*

Mitigasi yang dapat dilakukan pada kategori ini bukan berada pada aplikasinya, melainkan berada pada pihak pengembang aplikasi. Dimana pengembang aplikasi ketika mengembangkan aplikasi perlu memikirkan perencanaan juga pada sisi keamanan aplikasinya bukan hanya bagaimana aplikasi akan bekerja. Sehingga pengembang perlu memikirkan perencanaan yang matang untuk mengembangkan aplikasi.

4.3.5. Rekomendasi Mitigasi Kerentanan A05: *Security Misconfiguration*

Untuk memitigasi hal ini sangat mudah, hal ini hanya bisa diselesaikan melalui pembaharuan aplikasi itu sendiri, pengembang perlu melakukan *handle error* aplikasi yang baik sehingga detail *error* tidak sampai kepada pengguna. Karena informasi yang terlalu detail berbahaya karena dapat menyebabkan informasi sistem aplikasi bocor[29].

4.3.6. Rekomendasi Mitigasi Kerentanan A06: *Vulnerable and Outdated Component*

Untuk memitigasi hal ini menurut OWASP ada beberapa hal yang bisa dilakukan untuk memitigasi hal ini yaitu[30]:

1. Menghapus komponen atau *library* yang tidak digunakan.

2. Selalu memonitoring aplikasi *backend* maupun *frontend* menggunakan aplikasi seperti OWASP *depedency check* atau *retire.js.*

3. Menggunakan library yang berasal dari sumber resmi.

Hal ini harus dilakukan secara berkala terlebih lagi pada poin nomor 2 yaitu harus selalu di monitoring karena library atau perangkat lunak terus berkembang seiring berjalannya waktu.

4.3.7. Rekomendasi Mitigasi Kerentanan A07: *Identification and Authentication Failures*

Menurut OWASP hal yang bisa dilakukan untuk memitigasi hal ini adalah memperbaharui aplikasi dengan cara sesuai dokumen OWASP yaitu[31]:

1. Mengimplementasikan multi faktor autentikasi

2. Membatasi atau menambahkan waktu tunda ketika gagal login terjadi atau dalam kata artian menambahkan *rate limiting* pada aplikasi

Pada skenario ketika pengguna menggunakan komputer menurut OWASP *session management* harus dilakukan pada *backend* dan *session* aplikasi harus secara otomatis tidak berlaku ketika sudah *logout* pada aplikasi, ketika pengguna *idle* dan ketika memang *session* sudah saatnya tidak berlaku atau *absolute timeout* [31].

4.3.8. Rekomendasi Mitigasi Kerentanan A08: *Software and Data Intregity Failures*

Tidak ada yang harus di mitigasi dari ketegori ini karena pada aplikasi S.E.E.D.S untuk kategori ini aplikasi menggunakan pustaka pihak ke-3 dari sumber yang terpercaya.

4.3.9. Rekomendasi Mitigasi Kerentanan A09: *Security Logging and Monitoring Failures*

Mitigasi pada kategori ini bisa dilakukan cukup mudah yaitu dengan memperbaharui aplikasi dengan cara menghapus atau tidak melakukan *log* data sensitif pada aplikasi, ini sesuai dengan apa yang tertera pada OWASP [32].

4.3.10. Rekomendasi Mitigasi Kerentanan A10: *Server-Side Request Forgery(SSRF)*

Tidak ada yang harus di mitigasi dari ketegori ini karena pada aplikasi S.E.E.D.S untuk kategori ini aplikasi hanya mencatat data *url* ke *database* saja tidak sampai melakukan permintaan ke *url* hasil dari masukan pengguna.

4.4. Analisis Kerentanan OWASP Pada Aplikasi S.E.E.D.S

Tabel 4.10 Tabel Hasil Pengujian Aplikasi Berdasarkan OWASP

Kategori	percobaan gagal & Alat Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Rekomendasi Mitigasi
A01: *Broken Access Control*	percobaan: manipulasi hak akses pada token autentikasi alat pengujian: burp suite	Semua *resource* aplikasi dapat diakses oleh pengguna yang sah	Pada skenario ini ditemukan kerentanan	Pembaharuan aplikasi supaya aplikasi selalu melakukan cek *resource* yang diakses pengguna merupakan pengguna valid atau bukan
A02: *Cryptographic Failure*	percobaan: enkripsi sensitif data	Aplikasi menyimpan data di	Pada skenario ini	Pembaharuan aplikasi supaya aplikasi

	alat pengujian: *database gui/cli*	*database* atau saat transit secara aman	ditemukan kerentanan	melakukan enkripsi data sensitif
A03: *Injection*	percobaan: *SQL injection, XSS* alat pengujian: langsung melalui aplikasi	Aplikasi tidak melakukan eksekusi perintah yang diinputkan pengguna pada aplikasi	Pada skenario ini tidak ditemukan kerentanan	-
A04: *Insecure Design*	percobaan: OWASP nomor A01, A02, A05, A06, A07, A09 alat pengujian: dokumen perencanaan aplikasi	Pengembang melakukan perencanaan keamanan yang baik dalam pengembangan aplikasi	Pada skenario ini ditemukan kerentanan	Pengembang aplikasi kedepannya perlu memperhatikan keamanan aplikasi sebelum membuat aplikasi
A05: *Security Misconfiguration*	percobaan: eksposur informasi	Aplikasi tidak menggunakan konfigurasi	Pada skenario ini	Pembaharuan aplikasi supaya aplikasi

	sistem melalui error detail alat pengujian: langsung melalui aplikasi, *terminal/cmd*	yang *default* seperti menampilkan pesan *error* yang sangat detail atau aplikasi masih menyimpan akun testing atau akun bawaan	ditemukan kerentanan	melakukan tidak menampilkan pesan *error* yang detail
A06: *Vulnerable and Outdated Components*	percobaan: aplikasi *frontend* dan *backend* alat pengujian: OWASP depedency check	Aplikasi menggunakan pustaka pihak ke-3 yang terbaru dan tidak terdeteksi kerentanan pada pustaka tersebut	Pada skenario ini ditemukan kerentanan	Pembaharuan aplikasi supaya aplikasi melakukan *update* pada pustaka pihak ke-3
A07: *Identification and Authentication Failures*	percobaan: *Brute Force,* percobaan autentikasi pada komputer publik alat	Aplikasi tahan terhadap *brute force* dan bisa *logout* secara otomatis jika pengguna *idle* selama beberapa saat.	Pada skenario ini ditemukan kerentanan	Pembaharuan aplikasi supaya aplikasi menerapkan *rate limiting* dan/atau *two factor authentication* dan *logout* otomatis jika

	pengujian: *burp suite,* melalui aplikasi langsung			pengguna *idle* selama beberapa waktu
A08: *Software and Data Integrity Failures*	percobaan: aplikasi *frontend* dan *backend* alat pengujian: *terminal/cmd*, *browser dev tools*	Aplikasi menggunakan pustaka pihak ke-3 dari sumber terpercaya	Pada skenario ini tidak ditemukan kerentanan	-
A09: *Security Logging and Monitoring Failures*	percobaan: log fitur login, registrasi, undur diri alat pengujian: *terminal/cmd*	Aplikasi melakukan *log* disetiap aksi dan tidak menampilkan data sensitif saat *log*	Pada skenario ini ditemukan kerentanan	Pembaharuan aplikasi supaya aplikasi tidak melakukan *log* data sensitif
A10: *Server-Side Request Forgery (SSRF)*	percobaan: *Server-Side Request Forgery (SSRF)* alat pengujian: melalui	Aplikasi tidak melakukan permintaan pada *url* inputan pengguna atau melakukan validasi dari	Pada skenario ini tidak ditemukan kerentanan	-

	aplikasi langsung	*url* masukan pengguna		

Setelah dilakukan serangkaian pengujian berdasarkan dokumen OWASP, hasilnya pada aplikasi tidak lolos dokumen OWASP sebanyak 7 dari 10 kategori. Hal tersebut membuat aplikasi S.E.E.D.S ini dinyatakan memiliki kerentanan, karena sedikit apapun kerentanan tetap perlu diwaspadai karena bisa saja berimpact pada kepercayaan masyarakat luas terhadap suatu instansi apalagi didalamnya terdapat data-data yang bersifat sensitif atau pribadi [33][11].

BAB 5 KESIMPULAN DAN SARAN

5.1. Kesimpulan

Proses identifikasi kerentanan pada aplikasi S.E.E.D.S dilakukan dengan mengacu pada panduan *OWASP* Top 10, di mana pengujian mencakup sepuluh kategori kerentanan aplikasi web. Hasil pengujian menunjukkan bahwa terdapat tujuh kategori yang dinyatakan rentan, yaitu A01: *Broken Access Control*, A02: *Cryptographic Failure*, A04: *Insecure Design*, A05: *Security Misconfiguration*, A06: *Vulnerable and Outdated Components*, A07: *Identification and Authentication Failures*, serta A09: *Security Logging and Monitoring Failures*. Temuan ini menandakan bahwa metode pengujian berbasis OWASP efektif untuk mengidentifikasi area yang memerlukan perbaikan guna meningkatkan keamanan aplikasi. Rekomendasi mitigasi yang dapat diterapkan berdasarkan panduan OWASP meliputi:

1. A01: *Broken Access Control*: Menerapkan validasi dan pembatasan akses secara ketat agar pengguna hanya dapat mengakses sumber daya sesuai haknya.

2. A02: *Cryptographic Failure*: Menggunakan enkripsi yang kuat untuk melindungi data sensitif seperti informasi rekening pribadi.

3. A04: *Insecure Design*: Mengintegrasikan aspek keamanan sejak tahap perancangan aplikasi untuk meminimalkan potensi celah.

4. A05: *Security Misconfiguration*: Menghapus informasi sensitif dari pesan kesalahan agar tidak dimanfaatkan penyerang.

5. A06*: Vulnerable and Outdated Components*: Melakukan pembaruan rutin terhadap pustaka dan komponen pihak ketiga.

6. A07: *Identification and Authentication Failures*: Menambahkan mekanisme rate limiting, two-factor authentication, dan auto logout untuk meningkatkan keamanan autentikasi.

7. A09*: Security Logging and Monitoring Failures*: Menghindari pencatatan data sensitif pada log dan memperkuat pemantauan terhadap aktivitas mencurigakan.

5.2. Saran

Setelah melakukan penelitian, peneliti mampu memberikan beberapa saran yang dapat dijadikan pertimbangan untuk peneliti selanjutnya yaitu:

a. Pengujian dilakukan dengan menerapkan langsung rekomendasi mitigasi yang diberikan OWASP.
b. Pengujian dilakukan menggunakan metode yang sama yaitu OWASP tetapi menggunakan versi OWASP terbaru.

DAFTAR PUSTAKA

[1] C. M. Annur, “Indonesia Masuk 3 Besar Negara dengan Kasus Kebocoran Data Terbanyak Dunia,” https://databoks.katadata.co.id/teknologi-telekomunikasi/statistik/3f6a7dc0c5d0b7e/indonesia-masuk-3-besar-negara-dengan-kasus-kebocoran-data-terbanyak-dunia. Accessed: Oct. 28, 2024. [Online]. Available: https://databoks.katadata.co.id/teknologi-telekomunikasi/statistik/3f6a7dc0c5d0b7e/indonesia-masuk-3-besar-negara-dengan-kasus-kebocoran-data-terbanyak-dunia

[2] “LANSKAP KEAMANAN SIBER INDONESIA.”

[3] Hartono and O. W. Purbo, *Membangun dan Menguji Keamanan Website*. Yogyakarta: ANDI, 2022.

[4] R. Vallabhaneni, S. A. Vaddadi, S. E. Vadakkethil Somanathan Pillai, S. R. Addula, and B. Ananthan, “MobileNet based secured compliance through open web application security projects in cloud system,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 35, no. 3, pp. 1661–1669, Sep. 2024, doi: 10.11591/ijeecs.v35.i3.pp1661-1669.

[5] A. Choiriyah and N. Qomariasih, “Security Analysis on Websites Belonging to the Health Service Districts in Indonesia Based on the Open Web Application Security Project (OWASP) Top 10 2021,” in *Proceeding - International Conference on Information Technology and Computing 2023, ICITCOM 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 267–272. doi: 10.1109/ICITCOM60176.2023.10442816.

[6] “CWE Top 25 Most Dangerous Software Weaknesses,” https://cwe.mitre.org/top25/.

[7] “About the OWASP Foundation,” https://owasp.org/about/. Accessed: Sep. 20, 2024. [Online]. Available: https://owasp.org/about/

[8] “OWASP Top Ten,” https://owasp.org/www-project-top-ten/.

[9] P. Vishwakarma, “OWASP Top 10 vs SANS/CWE Top 25: Complete Security Vulnerabilities Guide [2024],” https://www.secopsolution.com/blog/owasp-top-10-vs-sans-25.

[10] T. Petranovic and N. Zaric, “Effectiveness of Using OWASP TOP 10 as AppSec Standard,” in *2023 27th International Conference on Information Technology, IT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/IT57431.2023.10078626.

[11] S. U. K. H. U. K. and M. I. R. A. Khan, “Systematic Literature Review on Security Risks and its Practices in Secure Software Development,” *IEEE Access,* vol. 10, Jan. 2022.

[12] B. Annie and F. Amber, “What is dynamic application security testing (DAST)?,” https://www.ibm.com/topics/dynamic-application-security-testing.

[13] “What is penetration testing?,” https://www.ibm.com/topics/penetration-testing.

[14] I. R. Dhaifullah, M. Muttanifudin, A. A. Salsabila, and M. A. Yakin, “Survei Teknik Pengujian Software,” 2022.

[15] M. G. and M. M.-T. D. Wang, “Information needs in bug reports for web applications,” *J. Syst. Softw.*, vol. 219, Oct. 2024.

[16] “Critical Applications,” https://www.sciencedirect.com/topics/computer-science/critical-application.

[17] M. A. Mohamed Mohideen *et al.*, “Behind the Code: Identifying Zero-Day Exploits in WordPress,” *Future Internet*, vol. 16, no. 7, Jul. 2024, doi: 10.3390/fi16070256.

[18] S. S. and R. R. P. S. L. Raju, “8 Use ModSecurity Web Application Firewall to Mitigate OWASP’s Top 10 Web Application Vulnerabilities,” *mplement. Enterprise. Cybersecurity with Open-source Software. Stand. Archi*, 2021.

[19] O. C. M. K. H. H. and A. D. O. Ben Fredj, “ An OWASP Top Ten Driven Survey on Web Application Protection Methods,” *pringer International Publishing*, 2021.

[20] A. P. S. P. and D. S. N. Chillur, “ Deep Analysis of Attacks and Vulnerabilities of Web Security.”.

[21] N. Anantharaman and B. Wukkadada, “Identifying the Usage of Known Vulnerabilities Components Based on OWASP A9,” *Iut. Infont. Conf. Emerg. Smart Comprmatics*, 2020.

[22] Z. Yin and S. U. J. Lee, “Security Analysis of Web Open-Source Projects Based on Java and PHP,” 2023.

[23] “OWASP WebGoat,” https://owasp.org/www-project-webgoat/.

[24] “Undang-undang (UU) Nomor 27 Tahun 2022 Pelindungan Data Pribadi,” PeraturanBPK. Accessed: May 20, 2025. [Online]. Available: https://peraturan.bpk.go.id/Details/229798/uu-no-27-

[25] “Authorization Cheat Sheet,” https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html#ensure-lookup-ids-are-not-accessible-even-when-guessed-or-cannot-be-tampered-with. Accessed: May 20, 2025. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html#ensure-lookup-ids-are-not-accessible-even-when-guessed-or-cannot-be-tampered-with

[26] “A01:2021 – Broken Access Control,” https://owasp.org/Top10/A01_2021-Broken_Access_Control/#how-to-prevent. Accessed: May 20, 2025. [Online]. Available: https://owasp.org/Top10/A01_2021-Broken_Access_Control/#how-to-prevent

[27] “A02:2021 – Cryptographic Failures,” https://owasp.org/Top10/A02_2021-Cryptographic_Failures/#how-to-prevent. Accessed: May 20, 2025. [Online]. Available: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/#how-to-prevent

[28] “A04:2021 – Insecure Design.” Accessed: May 20, 2025. [Online]. Available: https://owasp.org/Top10/id/A04_2021-Insecure_Design/#cara-mencegah

[29] “Error Handling Cheat Sheet,” https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html#sources-of-the-prototype. Accessed: May 20, 2025. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html#sources-of-the-prototype

[30] “A06:2021 – Vulnerable and Outdated Components,” https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/. Accessed: May 20, 2025. [Online]. Available: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

[31] “A07:2021 – Identification and Authentication Failures,” https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/. Accessed: May 20, 2025. [Online]. Available: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

[32] “Logging Cheat Sheet,” https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html#operational-use-cases. Accessed: May 20, 2025. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html#operational-use-cases

[33] Dr. J. Edwards, “Application Security,” in *Mastering Cybersecurity*, Berkeley, CA: Apress, 2024, pp. 139–171. doi: 10.1007/979-8-8688-0297-3_6.

LAMPIRAN

Lampiran 1

BERITA ACARA WAWANCARA

Dengan ini menerangkan bahwa mahasiswa yang bertanda tangan di bawah ini:

Nama : Lucky Tri Bhakti

NIM : 103012380500

Program Studi : S1 – Teknik Informatika

Univeritas : Universitas Telkom

Telah Melakukan Wawancara dengan :

Nama : Aditya Muhammad Noor Setiabudi

Jabatan : Kepala Urusan Riset TI

Perusahaan : Direktorat Pusat Teknologi Informasi Telkom University

Dalam rangka penyusunan Skripsi Teknik Informatika tahun akademik 2024-2025 yang berjudul "Analisis Kerentanan dan Rekomendasi Mitigasi Keamanan Pada Aplikasi Student Enrollment and Education Data System (S.E.E.D.S) Menggunakan Metode Pengujian Top 10 OWASP".

Bandung, 25 Mei 2025

Yang Diwawancara,

Aditya Muhammad Nur Setiabudi

Pewawancara

Lucky Tri Bhakti

Lampiran 2

Skenario	Alat pengujian	Hasil yang diharapkan
Mengakses halaman yang membutuhkan autentikasi tanpa melakukan autentikasi	Langsung melalui aplikasi	Aplikasi *redirect* ke halaman login

Manipulasi aplikasi dengan memalsukan token	*Browser devtools*	Aplikasi *redirect* ke halaman login
Modifikasi data pengguna orang lain	*Burp suite*	Aplikasi melakukan *response* 403 atau *response* gagal atau *redirect* kehalaman login
Manipulasi hak akses pada token autentikasi	*Browser devtools, webwolf owasp*	Aplikasi *redirect* ke halaman login

Tahapan pengujian:

Skenario mengakses halaman yang membutuhkan autentikasi tanpa melakukan autentikasi:

1. Buka aplikasi
2. Ubah *url* pada browser menuju ke /home
3. Cek apakah aplikasi *redirect* ke halaman *login* kembali atau tidak

Skenario manipulasi aplikasi dengan memalsukan token:

1. Buka aplikasi
2. *Login* menggunakan akun *dummy*
3. Klik kanan pada *browser* -> inspect -> klik tab storage -> klik localstorage
4. Lihat ada key “token”, *copy value* dari *key* tersebut lalu *logout*
5. Klik kanan pada *browser* -> inspect -> klik tab storage -> klik localstorage
6. Tambahkan key “token” lalu masukan token asal
7. Lalu akses kembali /home pada aplikasi

Skenario modifikasi data pengguna orang lain:

1. Buka aplikasi *burp suite*

2. Klik tab proxy

3. Klik open browser

4. Buka aplikasi pada

5. Akses halaman registrasi -> registrasi tahap 2

6. Klik intercept off supaya menjadi intercept on

7. Lakukan aksi simpan atau hapus

8. Cek pada *burp suite* permintaan ke aplikasi *backend*

9. Klik kanan pada permintaan tersebut lalu klik send to repeater

10. Ubah payload atau data yang dikirim yang sekiranya merupakan id atau pengenal dari data yang dikirim seperti uuid atau nik dan sebagainya

11. Klik send

Skenario manipulasi hak akses pada token autentikasi:

1. Buka aplikasi

2. *Login* menggunakan akun *dummy*

3. Klik kanan pada *browser* -> inspect -> klik tab storage -> klik localstorage

4. Lihat ada key “token”, *copy value* dari *key* tersebut lalu *logout*

5. Klik kanan pada *browser* -> inspect -> klik tab storage -> klik localstorage

6. Cari key “token” pada localstorage

7. *Copy value* tersebut

8. Buka aplikasi webwolf lalu klik menu JWT, ubah pada menu form “decode” “alg” jadi “none” dan payload ubah “role” menjadi “[‘admin-academic’]”

9.

10. *Paste value* yang di *copy* tadi ke form “encode”

11. *Copy token* baru pada form “encode” lalu *copy* ke localstorage aplikasi dengan *key* “token” dan *value* dari *key* yang baru di generate tadi

12. Buka halaman /home kembali

Lampiran 3

Skenario	Alat pengujian	Hasil yang diharapkan
Https downgrade	*burp suite*	Aplikasi meredirect *request*
Enkripsi sensitif data	*database gui/cli*	Aplikasi menyimpan data sensitif pada *database*

Langkah pengujian:

Skenario *Https downgrade*:

12. Buka aplikasi *burp suite*

13. Klik tab proxy

14. Klik open browser

15. Buka aplikasi pada “https://seeds.telkomuniversity.ac.id”

16. Akses halaman manapun

17. Klik intercept off supaya menjadi intercept on

18. Lakukan aksi pada halaman tersebut, jika *login* maka lakukan *login* jika pada tahap 1 maka lakukan pilih prodi dan seterusnya

19. Buka *burp suite* lihat permintaan dari aksi yang dilakukan

20. Klik permintaan tersebut klik kanan lalu klik send to repeater

21. Pada burp suite sebelah kanan atas klik ikon pensil untuk mengubah target

22. Ganti Https menjadi http klik lalu klik ok

23. Klik tombol send

24. Lihat *response* apakah *response* mengembalikan status 301 atau tidak

Skenario enkripsi data sensitif:

1. Buka aplikasi

2. Masukan data rekening

3. Cek data rekening pada *database* melalui *gui* atau *cli* apakah data tersimpah secara enkrispi atau *plain text*.

Lampiran 4

Skenario	**Alat yang digunakan**	**Hasil yang diharapkan**
SQL injection	*database gui/cli*	Perintah *sql* tidak tereksekusi
XSS	Langsung melalui aplikasi	Kode *javascript* tidak tereksekusi

Langkah pengujian:

Skenario *SQL Injection:*

1. Buat tabel “test” sebagai target untuk dihapus melalui *sql injection*

2. Buka aplikasi

3. Masukan *sql* “ test’; DROP TABLE test; -- ” ke semua form yang ada pada tahap registrasi 2 satu persatu.

4. Klik tombol simpan dan cek pada *database* melalui *gui/cli* apakah tabel “test” hilang atau tidak

Skenario *XSS:*

1. Buka aplikasi

2. Masukan “<script>alert(document.cookie)</script>” ke semua form yang ada pada tahap registrasi 2 satu persatu.

3. Simpan dan buka tab “review data” apakah kode *javascript* tereksekusi atau tidak

Lampiran 5

Untuk kategori A04: *Insecure Design* pengujian harus menguji semua kategori terlebih dahulu, lalu lakukan pengecekan ke dokumen perencanaan aplikasi apakah ada dokumen yang memuat mengenai bagaimana keamanan direncanakan jika tidak maka aplikasi termasuk rentan pada kategori ini. Contohnya jika pada dokumen tidak direncanakan bagaimana proses *logging* seharusnya maka aplikasi termasuk rentan pada kategori ini dan kategori A09: *Security Logging and Monitoring Failures*

Lampiran 6

Skenario	Alat pengujian	Hasil yang diharapkan
Eksposur informasi sistem melalui error detail	*Terminal/cmd*	Ditemukan kerentanan
CSRF	*Teks editor*	Tidak ditemukan kerentanan

Tahapan pengujian:

Skenario Eksposur informasi sistem melalui error detail:

1. Buka aplikasi lalu masuk ke menu tahap manapun.

2. Putuskan koneksi *database* mongo atau postgres:

Windows

1. Service: net stop MongoDB atau sc stop MongoDB
2. Manual run: Ctrl + C di terminal mongod

Linux

1. systemd: sudo systemctl stop mongod
2. init.d: sudo service mongod stop
3. manual run: Ctrl + C di terminal mongod

macOS

1. Homebrew: brew services stop mongodb-community
2. Manual run: Ctrl + C di terminal mongod

3. Lakukan aksi apapun pada aplikasi seperti simpan data, atau mengambil data lihat *response* pada aplikasi apakah mengembalikan data atau pesan *error* yang detail

Skenario CSRF:

1. Buat HTML seperti ini:

```
<!DOCTYPE html>

<html lang="en">



<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Voucher makan gratis!</title>

</head>



<body>

    <input type="text" placeholder="email">

    <button onclick="voucherGratis()">Klik untuk mendapatkan voucher makan gratis!</button>
```

```
  <script>
    function voucherGratis() {
      fetch("http://localhost:3004/akademik-admisi/registration/step-one/reset-registration-status", { method: "POST" })
        .then(res => res.json())
        .then(res => console.log(res))
    }
  </script>
</body>

</html>
```

ini berguna untuk menyerupai penyerang membuat aplikasi S.E.E.D.S palsu dan melakukan *request* ke *backend* S.E.E.D.S.

2. Simpan file html tersebut

3. Buka aplikasi masuk ke menu registrasi -> registrasi tahap 1 pilih prodi jika memang belum memilih

4. Buka file html yang sudah disimpan tadi pada browser

5. Refresh kembali aplikasi S.E.E.D.S apakah aplikasi kembali ke tahap 1 atau tidak.

6. Atau pada file html tadi klik kanan->inspect-> klik tab network

7. Lihat *response* dari permintaan ke *backend* S.E.E.D.S apakah mengembalikan status 200 atau tidak.

Lampiran 7

Skenario	Alat pengujian	Hasil yang diharapkan
Pengecekan aplikasi *frontend*	OWASP depedency check	Tidak memiliki pustaka pihak ke-3 yang usang
Pengecekan aplikasi *backend*		

Langkah pengujian:

1. Unduh aplikasi OWASP depedency check pada https://github.com/dependency-check/DependencyCheck

2. Akses direktori bagian *frontend* atau *backend* melalui *terminal/cmd*

3. Masukan perintah “path/to/dependency-check.sh --out . --scan .” untuk linux, untuk macos masukan “dependency-check --out . --scan .” untuk windows “path/to/dependency-check.bat --out . --scan .”. Ganti “path/to/” ke direktori dimana owasp depedency diinstall

4. Lihat hasil html yang dibuatkan oleh OWASP depedency check.

Lampiran 8

Percobaan	Alat pengujian	Hasil yang diharapkan
Brute Force	Ditemukan kerentanan	Aplikasi mengembalikan *response* 429 (*To Much Request)* atau mengirim kode OTP
Skenario autentikasi pada komputer publik	Ditemukan kerentanan (Selama 1 jam masih bisa diakses)	Aplikasi otomatis *logout* dan harus login *ulang*

Langkah pengujian:

Skenario *brute force:*

25. Buka aplikasi *burp suite*

26. Klik tab proxy

27. Klik open browser

28. Buka aplikasi

29. Klik intercept off supaya menjadi intercept on

30. Coba masukan email data dummy dan password sembarang

31. Buka burp suite kembali dan cek apakah ada permintaan ke *api* login

32. Klik kanan pada permintaan tersebut lalu klik send to intruder

33. Klik tab intruder dan cari dimana aplikasi mengirimkan payload passwortd

34. Block password dan klik “add $”

35. Klik payload pada burpsuite dan masukan password acak sebanyak mungkin, bisa cari pada google “common indonesia password” dan klik paste pada burp suite dan masukan password asli berdasarkan data akun dummy

36. Setelah itu klik start attack, jika aplikasi tidak mengembalikan return 429 atau mengirimkan OTP maka aplikasi rentang terhadap *brute force*

Skenario login pada komputer publik:

1. Buka *browser* akses aplikasi lalu login

2. Close browser atau close tab aplikasi

3. Tunggu selama 1 atau beberapa jam untuk menyerupai skenario pengguna membuka aplikasi pada komputer publik dan lupa *logout* aplikasi.

Lampiran 9

Skenario	Alat pengujian	Hasil yang diharapkan
Pengecekan aplikasi *frontend*	*browser devtools, terminal/cmd*	Tidak memiliki pustaka pihak ke-3 yang usang
Pengecekan aplikasi *backend*		

Langkah pengujian:

Pengecekan aplikasi *frontend:*

1. Buka aplikasi klik kanan -> inspect -> inspector

2. Buka pada bagian html tag head cek apakah ada css atau javascript dari sumber yang tidak terpecaya atau tidak. Contoh sumber yang terpecaya seperti: https://www.jsdelivr.com/ atau https://cdnjs.com/

Pengecekan aplikasi *backend:*

1. Akses ke server aplikasi *backend*

2. Masukan perintah “npm config get registry” jika registry bukan dari https://registry.npmjs.org/ maka bisa dianggap aplikasi mengunduh pustaka pihak ke-3 bukan dari tempat aman.

Lampiran 10

Menu	Alat pengujian	Hasil yang diharapkan
Login	*terminal/cmd*	Menampillkan log aksi namun tidak menampilkan data sensitif
Registrasi Tahap 1		
Registrasi Tahap 2		
Registrasi Tahap 3		
Undur Diri		

Langkah pengujian:

1. Buka setiap menu pada aplikasi

2. Lakukan semua aksi yang bisa dilakukan pada menu tersebut

3. Lihat *log* pada *terminal/cmd* apakah terdapat *log* jika iya apakah mengandung data sensitif atau tidak

Lampiran 11

Skenario	Alat pengujian	Hasil yang diharapkan
SSRF	*Browser devtools*	aplikasi memverifikasi masukan url atau aplikasi tidak melakukan permintaan berdasarakan masukan pengguna

Langkah pengujian:

1. Login pada aplikasi S.E.E.D.S

2. Klik kanan pada *browser* lalu klik "*inspect*"

3. Klik tab network

4. Masuk ke menu Registrasi -> Registrasi tahap 2

5. Scroll ke bawah ke bagian "jejaring sosial media yang aktif"

6. Masukan "http://localhost"

7. Lihat *response* pada *browser devtools* apakah aplikasi memunculkan *response* dari *localhost* atau tidak